THESIS FOR THE DEGREE OF LICENTIATE OF ENGINEERING

Towards Automated Experiments in Software Intensive Systems

DAVID ISSA MATTOS

Division of Software Engineering
Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden 2018

Towards Automated Experiments in Software Intensive Systems

David Issa Mattos

Technical report 178L
ISSN 1652-876X
Department of Computer Science and Engineering
Division of Software Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone + 46 (0)31-772 1000

Printed by Chalmers Reproservice,
Gothenburg, Sweden, 2018

Abstract

Context: Delivering software that has value to customers is a primary concern of every software company. One of the techniques to continuously validate and deliver value in online software systems is the use of controlled experiments. The time cost of each experiment iteration, the increasing growth in the development organization to run experiments and the need for a more automated and systematic approach is leading companies to look for different techniques to automate the experimentation process.

Objective: The overall objective of this thesis is to analyze how to automate different types of experiments and how companies can support and optimize their systems through automated experiments. This thesis explores the topic of automated online experiments from the perspectives of the software architecture, the algorithms for the experiment execution and the experimentation process, and focuses on two main application domains: the online and the embedded systems domain.

Method: To achieve the objective, we conducted this research in close collaboration with industry using a combination of different empirical research methods: case studies, literature reviews, simulations and empirical evaluations.

Results and conclusions: This thesis provides five main results. First, we propose an architecture framework for automated experimentation that can be used with different types of experimental designs in both embedded systems and web-facing systems. Second, we identify the key challenges faced by embedded systems companies when adopting controlled experimentation and we propose a set of strategies to address these challenges. Third, we develop a new algorithm for online experiments. Fourth, we identify restrictions and pitfalls of different algorithms for automating experiments in industry and we propose a set of guidelines to help practitioners select a technique that minimizes the occurrence of these pitfalls. Fifth, we propose a new experimentation process to capture the details of a trustworthy experimentation process that can be used as basis for an automated experimentation process.

Future work: In future work, we plan to investigate how embedded systems can incorporate experiments in their development process without compromising existing real-time and safety requirements. We also plan to analyze the impact and costs of automating the different parts of the experimentation process.

Keywords: Controlled experiments; A/B testing; multi-armed bandits; architecture framework; experimentation process; embedded systems; web systems

Acknowledgments

First, I would like to express my sincere gratitude to my supervisors. I would like to thank my main supervisor, Prof. Jan Bosch, for the patience of guiding me through this research while giving me the freedom to explore the topics that interest me. I would also like to thank my co-supervisor, Prof. Helena Holmström Olsson, for the invaluable support and feedback on my research. I could not have asked for better mentors.

Next, I would like to thank all at the Software Engineering Division for making it a great place work and thank all the companies and individuals that we worked with, the companies at the Software Center, Sony Mobile and Microsoft's Analysis and Experimentation team.

Finally, I am grateful for my parents, my brother and sister and my grandfather, for all the encouragement to pursue this dream. I would like to express my deepest gratitude to my life companion Erika, without your patience, support, love and care, I would not have been able to get to complete much of what I have done.

This work was partially supported by the Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP), funded by the Knut and Alice Wallenberg Foundation.

List of Publications

This thesis is based on the following publications:

[A] D. I. Mattos, J. Bosch, and H. H. Olsson, "Your System Gets Better Every Day You Use It: Towards Automated Continuous Experimentation," in *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2017, pp. 256–265.

[B] D. I. Mattos, J. Bosch, and H. Holmström Olsson, "More for Less: Automated Experimentation in Software-Intensive Systems," in *The 18th International Conference on Product-Focused Software Process Improvement*, 2017, pp. 146–161.

[C] D. I. Mattos, J. Bosch, and H. H. Olsson, "Challenges and Strategies for Undertaking Continuous Experimentation to Embedded Systems: Industry and Research Perspectives," in *Lecture Notes in Business Information Processing*, vol. 314, 2018, pp. 277–292.

[D] D. I. Mattos, E. Mårtensson, J. Bosch, and H. H. Olsson, "Optimization Experiments in the Continuous Space: The Limited Growth Optimistic Optimization Algorithm", *to appear in the Proceedings of the 10th International Symposium on Search-Based Software Engineering,* 2018, pp. 1-15.

[E] D. I. Mattos, J. Bosch, and H. H. Olsson*,* "Multi-armed bandits in the Wild: Common Pitfalls in Online Experiments" *in submission to an international software engineering journal,* 2018, pp.1-22.

[F] D. I. Mattos, P. Dmitriev, A. Fabijan, J. Bosch, and H. H. Olsson, "An Activity and Metric Model for Online Controlled Experiments" *to appear in the Proceedings of the 19th International Conference on Product-Focused Software Process Improvement,* 2018, pp. 1-16

Personal Contribution

I was the main contributor regarding to the design, planning, execution and writing of this thesis and all publications listed in the List of Publications.

Table of Contents

1 Introduction

Understanding the customer preferences and needs has become of strategic importance for software development companies to survive and grow. Today, software development companies use the collected data to assist their decision-making and gain insights into customer preferences. Customer data enables companies to timely adapt their business strategies and gain a competitive edge by delivering better products and services to their customers [1].

From web-facing companies to embedded systems, software development organizations collect data from multiple sources to help them in their decision-making and development process [2]. However, many software development organizations find themselves with an overwhelming amount of unreliable and unstructured data. One of the primary challenges for these companies is to develop a trustworthy data management process, from collection to analysis, to support their decisions and to help them solve problems technical problems, such as crashes and unexpected software behavior and problems relevant to customers [3] such as if the current implementation of a feature is satisfactory and is being used. Even technically correct, software functionalities might not be relevant and fail to deliver value to the customers or they can deteriorate the customer experience [3]–[5].

During the development of software, companies aim to prioritize the development of software features and functionalities that deliver value and that are relevant to the customers [3], [5]–[8]. The development of a full feature from conception to user deployment can result in inefficiency and opportunity cost if it does not have a confirmed value. In a competitive software market, the prioritization of software features and services that deliver value to customers is critical for the success of every company [3], [4].

Previous studies show that, often in the development of systems, the prioritization of a feature is driven by guesswork, previous experiences and beliefs of those involved in the feature selection process [3], [4], [6], [9]. However, this prioritization process might not be aligned with the user preferences, behaviors or with the company's business goals. This mismatch is leading some software industries to look towards a more systematic way to deliver value in their development approaches [6], [10] to avoid disruption in an increasingly competitive industry.

Early customer feedback is important to allow the companies to make adjustments and redirect the software development from activities that are not delivering value to other development activities, that are either delivering more value or better aligned with the company's goals [11]. In order to make more informed decisions, these software companies rely on several qualitative and quantitative customer feedback techniques such as: surveys, interviews, participant observations, prototypes, mock-ups, feature usage, product data and support data to determine and validate a feature value [12].

However, not all of the mentioned customer feedback techniques are adequate for fast-iterating software development, a large and diverse user base, modern software development approaches such as Agile development [13], [14] and other technological advances such as continuous integration [15] and deployment [16]. For example, although interviews and participant observations can generate valuable insights on the user behavior, these techniques are expensive, they can take several months, and they are based on a limited number of participants, which means that these methods generate little amount of data. Even observational studies with a sample of hundreds of people can be biased towards a user segment if the software is used by several thousand users or the study might not have power enough to detect a small effect size

that still has practical significance [17]. Therefore, companies need additional techniques to collect a larger amount of trustworthy and unbiased structured data to help in the development and decision-making of not only larger modifications with a big impact but also incremental changes in smaller functionalities. Companies are investing in the collection and analysis of post-deployment data to improve and optimize their current products, to drive innovation in features and to generate insights on the user preferences [18], [19].

A controlled experiment in software engineering is an empirical method where an observer can test a hypothesis by manipulating one or more factors on randomized subjects, while keeping the other factors constant, and observe effects on the outcome variables [20]. This technique allows the observer to make causal inferences regarding the manipulated factors and the outcome variables. Combining post-deployment data combined with controlled experiments allows software companies to make causal inferences between changes in their software and user-behavior. A/B testing stands as one of the key techniques to run control experiments in online software systems. A/B testing refers to the technique where a company compares, in a controlled experiment setup, the existing system (variation A) against the system with a modification (variation B). If the experiment is executed properly the company can establish a causal effect between the modification in the system and the outcome metrics. Establishing this causal effect is important for companies to redirect their development efforts to modifications that are proved to deliver value. Several publications and reports from companies such as Microsoft, Facebook, Google, and LinkedIn, among many others [17], [21]–[23], report the competitive advantage that online controlled experiments (such as A/B testing) deliver [24].

As companies start to verify the benefits of combining post-deployment data with controlled experiments, they start to adopt experimentation as a standard development practice [6], [21]–[23], [25]. With the increasing number of simultaneous experiments and parts of the organization running [26], it is becoming unfeasible for the companies to grow the size of the organization at the same rate [27]. This situation is leading companies to look for different techniques that can automate the experimentation process and reduce the cost and time of each experimental iteration [23], [28]–[31]. In this scenario, machine learning, artificial intelligence and self-adaptation techniques can aid the experiment organization to automate this process and run more experiments at a lower cost [32], [33].

This thesis investigates the development and usage of automated experiments in software systems from three main perspectives: (1) the software architecture of an automated experimentation system, (2) the usage of machine learning algorithms, such as multi-armed bandits for automated experiments and (3) the experimentation process. This thesis looks at the automated experimentation problem focusing in two main application domains: the web-facing and embedded systems domain.

The web-facing domain corresponds to companies that develop primarily websites and online mobile applications. Companies in this domain have extensive expertise in online controlled experiments and has made several contributions in the development of automated online experiments [23], [28], [34]. Web-facing companies see automated experiments as a way to run more experiments at a lower cost per experiment and pipeline the experiments in a systematic and trustworthy process less prone to human mistakes [35].

The embedded systems domain (automotive, telecommunication and consumer electronics) is still in its early stages of running experiments, but it sees online experiments as one of the approaches that can increase their competitiveness. As discussed later in this thesis, some embedded systems companies see automated experiments as a way to lower the costs and the barrier to start running their first experiments and scale their process from ad-hoc experiments to a more systematic process.

This thesis provides five main contributions. First, it provides an automated experimentation framework together with a discussion on the necessary architectural qualities and on the architectural design decisions. Second, it identifies the key challenges faced by embedded systems companies when adopting continuous experimentation and proposes a set of strategies to address these challenges. Third, it develops a continuous-armed bandit algorithm suitable for online experiments and uses this algorithm in an instantiation of the previously developed framework in an existing commercial product. Fourth, it identifies a set of restrictions and pitfalls of multi-armed bandit algorithms applied to online experiments and we identify strategies used by practitioners to avoid these pitfalls and we propose a set of guidelines to help practitioners select a technique that minimizes the occurrence of these pitfalls. Fifth, this thesis proposes an experimentation process framework that captures the details of a trustworthy experimentation process.

This thesis is organized as follows. Chapter 1introduces the automated experimentation to the topic as well as presenting the goals of this Ph.D. research, the research questions and the key contributions. Chapter 2 provides a common background discussion to the rest of the thesis, discussing how online controlled experiments are conducted, the different experimentation process models, self-adaptive architectures and a review on multi-armed bandits algorithms. Chapter 3 discusses the research method, as well as the research questions and the motivation for using each of the methods among the studies.

Chapter 4, 5, 6, 7 and 8 are based on the publications A to F and constitute the key contributions of this thesis. Chapter 4 discusses architectural aspects of an automated experimentation system and proposes an architecture framework. Chapter 5 discusses the use of controlled experiments in the embedded systems domain and suggests that automated experiments might be a key technique for faster adoption of experiments in this domain. Chapter 6 proposes a new continuous-armed bandit algorithm and use it in an industrial context with an instantiation of the architecture framework proposed in Chapter 4. Chapter 7 discusses several pitfalls and restrictions of using multi-armed bandit algorithms in online controlled experiments and presents strategies to overcome these pitfalls. Chapter 8 discusses the experimentation process from a broader view than just the execution part. A broader understanding of the experimentation process allows us to investigate automation in experiments in different parts of the experimentation pipeline. Chapter 9 concludes the thesis with a discussion of the main results and a discussion for future work.

2 Background

This chapter reviews concepts on topics related to online controlled experiments, multi-armed bandits, software architectures and adaptation. Online controlled experiments are relevant for all discussions presented in Chapters 4, 5, 6, 7 and 8. Multi-armed bandits algorithms are used in online experiments and are used to automate the experiment execution. This discussion is relevant for Chapters 4, 6 and 7. The discussion on software architecture and adaption is relevant to Chapter 4 as it analyzes several architectures from different approaches to develop an architecture framework that supports automated online experiments.

Section 2.1, outlines the basic concept of online controlled experiments and discusses different traditional techniques used in online experiments together with examples, such as A/B/n, factorial and fractional experiments. This section also discusses the different experimentation models used in research and practice. These models present how an online controlled experiment process can be adopted by a development organization.

Section 2.2 presents the multi-armed bandit problem formulation, outlines some basic algorithms such as the ϵ-greedy, softmax and UCB1, and possible extensions. Multi-armed bandit algorithms are classified in the reinforcement learning umbrella and are used to leverage existing experimentation systems to automate the experiment execution.

Section 2.3 outlines the architectural concepts used in this thesis, software architectures for adaptation and different adaptation approaches. Self-adaptation research already developed several concepts that can be applied to the automated experimentation problem, including multiple software architectures that addresses different parts of this problem. Implementation of both automated and non-automated experimentation systems is often proprietary and limited in research. To develop an automated experimentation system, this thesis bases the discussion of the software architecture and the design decisions of an automated experimentation system in available research results on architectures for self-adaptation.

2.1 Online controlled experiments

This subsection details online controlled experiments techniques, such as how A/B experiments and further extensions are conducted and analyzed, and the existing experimentation process models.

2.1.1 Online controlled experimentation techniques

2.1.1.1 A/B and A/B/n experiments

A/B experiments are a group of techniques based on the design of experiments [36], where users are randomly assigned to different variants of the product. The control variant is the current system without any modifications and the treatment variants are the current system with a modification X_i, with $i \in \{1,2 \dots n\}$, where n represents the number of different variants. The modification X_i can be modifications in existing functionalities of the system (e.g. a different set of parameters for each variation), or the system with a new feature or functionality (e.g. different implementations of new features). All variations are properly instrumented and send usage data to the research and development organization. Metrics to evaluate the system are grouped and computed based on the variation of the system. The different variations are randomly assigned to different users, and, after a predetermined period of time, the metrics for each variation are statistically compared. If the only consistent difference between the experiments' variants is the modification X_i, and the randomization assumptions hold true, the research and development organization can establish a causal relationship between the

modification in the system and the change in the metrics between the different variations. Statistical analyses in both frequentist and Bayesian paradigms are used to compute the difference between the metrics [31], [37].

An A/B experiment refers when there is only one treatment variation and A/B/n refers when there are multiple levels for the same factor, i.e. n treatment variations. In A/B/n experiments, users are only exposed to one variation at a time and there are no interaction effects

An assumption is that the metrics for all variations should be comparable so an unbiased decision can be made, that the randomization is unbiased (all users have equal probability to be assigned to each variation), and that external factors (that influence the user behavior) are evenly distributed between all variations. If any of these assumptions do not hold for the entire duration of the experiment, the causal relationship between variations and metrics is compromised and can lead to wrong conclusions.

As the users are not in a controlled environment, the influence of external factors can be higher than the difference caused by the variations. To detect such differences, the research and development organization should use high confidence levels and high-power experiments. However, these restrictions lead to experiments with a high number of users in each group. This high number of users proves to be challenging to run trustworthy A/B experiments in many small- to medium-sized companies. Additionally, collecting such an amount of data can take between weeks and months [28], [38], even in large companies. The time to run even a single experiment can negatively impact the decision-making process and time to deploy a new feature in most software organizations.

To overcome the limitations in the number of experiments that companies can run simultaneously, along with limitations in the number of users required to reach a conclusion, software organizations have started to run other experimentation techniques.

A/B experiment example: As an example, suppose a company that produces car navigation software for mobile devices wants to evaluate if users enjoy using their new turn-by-turn navigation system. The company decides to use as metric the number of trips made with system on and volume higher than 10%. Each user is assigned randomly with equal probability to one of the two variants: the old system or the new system. For all trips the turn-by-turn navigation comes activated by default. The company determines a confidence level, a power, and a minimum effect size that it hopes to achieve with a new system. Based on these parameters and the user base, the company calculates the time to complete the experiment. After the data is collected, the experiment is analyzed and the best variant (or the control if it wasn't detected any statistical significance) is assigned to all users. The result of the experiment might have a deeper impact in the organization, as the new system might take over the old one, or be abandoned.

A/B/n experiments: Suppose a company wants to evaluate how 3 new ad recommendation algorithms (the current A and the new algorithms are B, C and D) move a click metric. Each user is assigned to one of the variants with equal probability (0.25) and will see only this variant throughout the experiment. Based on the defined confidence level and power, the experiment will run for a predetermined time. After the data is collected, the experiment is analyzed and the best variant (or the control if it wasn't detected any statistical significance) is assigned to all users.

2.1.1.2 Full-factorial experiments

This technique refers to running experiments with two or multiple levels for two or more factors. This means running experiments considering two changes in the system at the same time. This type of experiment considers that the different variations can have interaction effects.

The advantage of this method is to try different parts of the system at the same time and detect and learn interaction effects between different changes. This can lead to a better understanding of the system and of the users [37]. A disadvantage of this method is that the number of users necessary to run a full-factorial experiment is proportional to the number of different experimental groups.

Suppose a company wants to evaluate how 3 modifications in the layout of the entertainment system of a car: change the number of rows (R) of items displayed in the main screen (R=2 or R=3), the number of columns (C) of items displayed in the home screen (C=2 or C=3) and the presence (B=on) or absence (B=off) of a navigation bar (B) in the home screen. Sequential testing (first test the change R then the C then the B) cannot be used because interaction effects can appear between the experimental In this case, we use a full factorial design with 8 simultaneous variants (R=2,C=2,B=off is the control; R=2,C=3,B=on; R=3,C=2,B=on; R=3,C=3,B=off; R3,C=3,B=on; R=3,C=2,B=off; R=3,C=2,B=on and R=3,C=3,B=off). In this case interaction effects are also computed and are part of the analysis.

2.1.1.3 Fractional factorial experiments

The design of experiment literature [36] provides an alternative to full factorial experiments, where the number of experimental groups is reduced. Empirically, higher order interactions (more than two factors) are rare. Therefore, during the design of the experiment, factors can be confounded to these higher order interactions, reducing the number of necessary experimental groups. One noticeable disadvantage of this method is that if the assumption that the higher order interactions are rare is not true, the whole data collection and analysis is compromised. Kohavi et al. [37] provide further discussion on why this method is not ideal for online experiments.

Continuing with the example of the full factorial experiment. If we consider that the interaction effect between all three experimental changes is low (in this case there should be a well-based reason to believe that), we can confound this interaction with one of the factors reducing the number of experimental variants by half (R=2,C=2,B=off is the control; R=3,C=2,B=on; R=2,C=3,B=on; R=3,C=3,B=off;).

2.1.1.4 Overlapping experiments

Overlapping experiments are one-factor experiments that are launched simultaneously. In contrast to factorial designs, this type of experiment should not have any interactions, either because they refer to different parts of the system that should not affect each other (separation of experiments in design), or because users that are exposed to one experiment are not exposed to a second experiment that might interact (separation of experiments in users). In this case, in contrast to fractional designs, the research and development organization can estimate the interaction between factors in different experiments if there is a significant overlap. This approach is often recommended instead of fractional designs [37]. However, as the number of experiments in the organization scale, overlapping experiments might prove challenging, requiring complex coordination systems [21], [23]. In this technique, multiple experiments are launched together, but as they are separated in either the functionality under experiment or by exposing to different the number of users that are assigned to more than one experiment that can have interaction is low and the interaction can also be estimated. This approach is often used by companies that run several hundred of simultaneous experiments and have a very large user base.

2.1.1.5 Experimentation metrics

The choice of metrics for online experiments can have a considerable impact in organization, and it is often considered to be one of the key challenges [39] in online experiments. Determining good metrics is a significant problem as the metrics often represent abstract

business concepts (e.g. user engagement, user satisfaction), influence the alignment of the experiment organization with the business goals, and impact upon the number of users required for each experiment through the concept of sensitivity [40]. Previous research provides information on good metric qualities and characteristics [40], different types of metrics and how to evaluate them [39], and discussion on metrics' lifecycle and how they can impact upon business goals [39], [41].

2.1.2 Experimentation process models

This subsection presents the existing frameworks and models for online experimentation that capture the experimentation process and aid practitioners in establishing an experimentation pipeline during product development. The frameworks focus on innovation and learning outcomes, and optimization experiments are supported as an iterative A/B testing experiment procedure. MAB algorithms for online experiments that are built on top of these frameworks hide important assumptions of these algorithms. A violation of these assumptions can lead to wrong conclusions that might have negative impact on the business.

2.1.2.1 The Build-Measure-Learn model

The Lean Startup methodology [42] proposes an approach for companies to continuously and systematically innovate from a startup perspective. The methodology employs a Build-Measure-Learn cycle to ensure that the software development is aligned with the customer's wishes. One of the key aspects of this Build-Measure-Learn cycle is to run scientific experiments to validate customer needs and to ensure that the product is aligned with these needs. The build phase reinforces the use of a minimum viable product as an approach to fast steer the direction of the product roadmap in a startup environment. The measure phase emphasizes the needs of instrumentation in the products to measure users' and systems' behavior. The learn phase uses collected post-deployment data to understand and learn movements in hypothesis metrics. This methodology describes a general experimentation process similar to A/B experiments for learning and innovation.

2.1.2.2 The ESSSDM model

The Early Stage Software Startup Development Model (ESSSDM) [43] proposes an operational support, based on lean principles, for practitioners to investigate multiple ideas in parallel and scale their decision-making process. The model consists of three steps. The first is the idea generation, in which the startup (or the existing company) generates ideas to expand their product portfolio. The second is the prioritization of the potential ideas in a backlog. The third is the systematic validation funnel using a Build-Measure-Learn loop similar to the Lean Startup methodology. In this step, multiple ideas can be investigated and validated in parallel. The funnel is divided in four stages: the validate problem, the validate solution, the validate the minimum viable product on a small scale, and the validate the minimum viable product on a large scale. This model supports the use of A/B experiments for learning and innovation in a similar manner as the Build-Measure-Learn model.

2.1.2.3 The Stairway to Heaven model

The Stairway to Heaven [44] refers to a pattern that most companies follow in their evolution process from waterfall development to research and development as an experiment system. The model starts with companies going from a traditional development pattern such as the waterfall to agile, continuous integration, continuous deployment, until the R&D as an experiment system. In this final step, companies start to run continuous A/B experiments, not only for innovation and learning (as in the models based on startups), but also for the optimization of their systems. In this optimization stage, companies want to improve their experimentation process and start to look at and run other experimentation procedures, as multi-armed bandit algorithms.

2.1.2.4 The QCD model

The QCD model (Quantitative/qualitative Customer-driven Development) [45] explores the continuous validation of customer value instead of relying on up-front requirement specification. The QCD model treats requirements as hypotheses that need customer feedback for validation in the beginning of the development process. All hypotheses are gathered in a hypotheses backlog, where they are prioritized and selected for evaluation. In the validation cycle, the selected hypothesis is evaluated through both quantitative and qualitative feedback. If the hypothesis is not confirmed through the evaluation techniques, it can be either refined in another hypothesis for a future iteration or abandoned. This model provides a higher-level experimentation process abstraction. It considers both qualitative and quantitative data analysis methods (as A/B and multi-armed bandit experiments).

2.1.2.5 The HYPEX model

The HYPEX (Hypothesis Experiment Data-Driven Development) model [46] is a development for companies aiming to shorten the feedback loop to customers. Instead of spending engineering efforts on large pieces of functionality that were not validated by customers, the HYPEX model reinforces the need for an iterative and incremental approach. The model divides the development process into six steps: (1) generation of a feature backlog, (2) feature prioritization and gap specification, (3) implementation of a minimum viable feature (MVF), (4) analysis and comparison of the actual behavior with the expected one, (5) generation of hypotheses to explain the actual behavior of the MVF, and (6) deciding if the feature should be abandoned, iterated once more, or if it should be considered completed. This model supports A/B experiments focused in feature innovation and learning; however, the model does not capture specific details to support A/B experiments or how it can be extended to multiple variations, or for optimization experiments as in MAB and A/B/n experiments.

2.1.2.6 The RIGHT model

The RIGHT (Rapid Iterative value creation Gained through High-frequency Testing) [3], [4] is a model for driving A/B experiments in a startup environment.. The RIGHT process model uses the Build-Measure-Learn loop to help a startup company to achieve the company's vision through continuous experiments. Hypotheses that implement business strategies are generated and prioritized, minimum viable features or products are implemented and instrumented, and data are collected. The analysis of the collected data helps the decision-making process in a similar manner to the HYPEX model, where decisions to continue iterating, abandoning, or moving on to the next cycle are made. The RIGHT model describes a high-level experimentation process that can be used in innovation and learning A/B experiments. Optimization experiments are not considered in this process, as it does not specify how to establish a scalable and trustworthy experimentation process that can help companies to achieve the last step in the Stairway to Heaven model, i.e. the R&D as experiment systems.

2.1.2.7 The Experimentation Evolution Model

Fabijan et al. [26] describes the Experimentation Evolution Model, based on experimentation at Microsoft. This model analyzes how teams scale their experimentation from a few experiments to a data-driven organization. The model divides this evolution into four steps: crawl (teams are running and setting their first experiments), walk (teams already run a few experiments and determining metrics and experimentation platforms), run (the teams run several experiments and iterate quickly to identify effects of experiments on the business) and fly (experiments are the norm for every change to any product). Each of these phases is discussed in three different perspectives, the technical, the organizational, and the business perspectives.

2.2 The Multi-Armed Bandit problem and algorithms

This section describes the multi-armed bandit (MAB) problem, provide a common background notation for the MAB problem, a description of three main algorithms: the ϵ-greedy, Softmax and UCB1 and a discussion of further extensions.

2.2.1 The MAB problem formulation

MAB (also known as K-armed bandit) is a class of problems classified inside the umbrella of reinforcement learning, that sequentially experiment with the goal of producing the greatest reward [47]. This experiment process deals with the exploration and exploitation trade-off. The problem statement and its name come from the parallel of the gambler problem facing a row of K slot machines, also called one-armed bandits. In its simpler formulation, the gambler has a limited number of tries to play, the number of arms is finite (equals to K), the arms are independent of each other (the reward of one arm does not influence that of the other arms), and each arm has a stationary stochastic distribution over time (the mean-reward probability distribution does not change with time).

Several solutions for the MAB problem were proposed in a wide range of applications, conditions and assumptions that go beyond the simpler formulation presented previously. Some applications of MAB are in clinical trials [48], financial portfolio design [49], [50], search engine [51] and news [52] recommendation systems and online experiments [47]. Interest in MAB for online experiments started when companies faced some of the aforementioned limitations of A/B testing and other experimental designs. The research literature suggests MAB algorithms as being alternative approaches to A/B experiments [53], [54].

The general problem can be formalized as [34]:

$$a = \pi(\delta), \text{Arm } a \in \{a_1 \dots a_K\} \quad (1)$$

$$y = r(a, \delta'), \text{Reward } y \in \mathbb{R} \quad (2)$$

where a is the arm selected, K is the number of arms, π is the user-defined policy function to balance the exploration of arms and the exploitation of the best arm so far, δ is and $\delta'$are noise variables, y is the measured reward and r is the unknown reward function for the selected arm. The MAB algorithms are the construction of the user-defined policy π to select the arm. In online experiments, each arm is a possible variation of the system. This can be different implementations of a feature, different parameters, or different changes. The reward y is the (user-)metric used to measure the difference between the arms. The reward metric should have a positive direction (the higher value of y is considered better than a lower value). Although the reward can be considered in a continuous real space, several MAB algorithms restrict the range to the interval $y \in [0, 1]$.

Most algorithms are formulated in terms of the concept of regret. Regret is a comparison of the cumulative mean reward of the algorithm and the expected reward of playing the optimal arm.

$$Regret(t) = r(a^*) \cdot t - \sum_{s=1}^{t} \mu(a_s) \quad (3)$$

where $\mu(a)$ is the mean reward of the arm a over the time and a^* is the optimal arm:

$$a^* = \max_{a \in \{a_1 \dots a_K\}} \mu(a) \quad (4)$$

The literature on multi-armed bandits explores different formulations for the regret function [53], which is beyond the scope of this work.

Next, we describe different MAB algorithms. These algorithms were chosen because different implementations and more complex algorithms are based on strategies similar to those present

in these algorithms. An extensive review of different MABs and extensions is provided in the surveys by Burtini et al. [53] and Kuleshov and Precup.[54].

2.2.2 Explore-First algorithm with parameter N

The simplest algorithm for the MAB problem is the explore-first strategy with parameter N. This strategy consists of uniformly exploring all the arms equally for a finite horizon of NK play, where each arm is played N times, and K is the number of arms. After NK plays, the arm with the highest average reward is selected. Ties between arms are broken arbitrarily.

Exploration: In this step, all arms are selected with a uniform probability during the first N rounds.

$$a = U(a_i), \text{Arm } a \in \{a_1 \dots a_K\} \tag{5}$$

In this step, the arm with the highest average reward is selected:

$$a^* = \max_{a \in \{a_1 \dots a_K\}} \mu(a) \tag{6}$$

Exploitation: In this step, all the remaining rounds are played with the best arm a^*.

This strategy is often compared to A/B/n testing. However, in A/B/n the selection of the best arm follows a statistical procedure with a fixed significance level. The strategy of selecting the highest reward without computing the significance level or minimizing type I errors is, in this text, called the naïve MAB arm selection.

2.2.3 The ϵ-greedy algorithm

This algorithm exploits the best arm (the arm with the highest mean reward) with probability $1 - \epsilon$ and selects, with probability ϵ, an arm at random (including the current arm with the highest mean reward), uniformly and independently of the arm-value estimates [18]. This solution is equivalent to the one-state Markov decision process problem [53].

In terms of the exploration and exploitation trade-off, this algorithm explores $\epsilon \cdot t$ while exploits $(1 - \epsilon) \cdot t$ of the time.

This strategy can be represented as:

$$a = \begin{cases} \max_{a \in \{a_1 \dots a_K\}} \mu(a), \text{with probability } 1 - \epsilon \\ U(a), \text{Arm } a \in \{a_1 \dots a_K\}, \text{with probability } \epsilon \end{cases} \tag{7}$$

The case where $\epsilon = 1$ reduces the ϵ-greedy algorithm to the ϵ-first strategy.

2.2.4 The Softmax algorithm

In a similar manner to the ϵ-greedy algorithm, the Softmax algorithm also exploits the best arm but explores by ranking and weighting the other arms according to their value estimates. The Softmax algorithm used in this work uses the Boltzmann function to give each arm its probability [55]. The Softmax algorithm has the ability to assign a small probability to the worst arms [56].

Each arm has a probability of being selected depending on the arm mean reward according to:

$$P(a_i) = \frac{e^{\frac{\mu(a_i)}{\tau}}}{\sum_{i=1}^{K} e^{\frac{\mu(a_i)}{\tau}}} \tag{8}$$

The Softmax algorithm uses a parameter τ called temperature. High temperatures cause the arms to be nearly equi-probable, while $\tau \to 0$ makes the arm selection to become the same as the greedy algorithm. Both the Softmax and the ϵ-greedy algorithms have one parameter for

tuning. However, setting τ requires a deeper understanding of the likelihood of each arm value [56].

2.2.5 The UCB1 algorithm

The Upper Confidence Bound (UCB), or worst-case regret bound, algorithm is a different algorithm implementation that not only explores how much reward each arm receives, but also the confidence in each arm. The algorithm balances the exploration/exploitation tradeoff by selecting the arm that has the highest empirical reward estimate. The aim of this algorithm is to minimize regret in making an arm decision [53].

The user-defined policy can be written as:

$$a(t) = \arg\max_{i=1\dots K} \left(\hat{\mu}_i + \frac{\sqrt{2 \ln t}}{n_i} \right) \quad (9)$$

where $a(t)$ represents the arm selected at the time t.

This user-defined policy does not use randomization in the arm selection, but it rather depends on the randomization of the reward. Additionally, the UCB1 does not have free parameters, making it easier to deploy without prior tuning. However, one of its assumptions is that the reward value should lie between 0 and 1 [55].

2.2.6 Further extensions

The MAB and the exploration/exploitation dilemma are widely studied and have several extensions and improvements. The first extension to be considered is the case of contextual bandits. When prior information of how the reward works (e.g. if it follows a known pattern or if it behaves differently from groups of users) is available, an exogenous context variable (x) representing this information can be included in the problem formulation:

$$a = \pi(x, \delta), \text{Arm } a \in \{a_1 \dots a_K\} \quad (10)$$

$$y = r(x, a, \delta'), \text{Reward } y \in \mathbb{R} \quad (11)$$

A second extension is the use of multiple rewards. As formulated, the MAB problems utilize a one-dimension reward. In the case of optimizing for multiple parameters, also known as the multi-objective MAB, one approach is to utilize Pareto relationships to drive the optimization process [57].

Other extensions utilize Bayesian learning, Thompson sampling, stochastic and nonstationary bandits, and also MABs where the number of arms is considerably larger than the exploration space, or belong to a continuous space [53]. However, several of those extensions are research implementations, described only through simulations and datasets, and have not yet reached industrial use.

2.3 Software architecture and adaptation

This section discusses concepts on software architecture and self-adaptation

2.3.1 Software architecture and design decisions

Software architecture is an abstraction of the set of structures needed to reason about the properties, the elements and the relationships among them in a system [58]. The importance of architectural patterns, the relation between the components and their connectors, the use of well-defined modules, separation of concerns and achieving of quality attributes to well-known patterns and tactic are generally recognized to lead to better control over the design, development and evolution of the systems [59].

However, it is also recognized that some software architecture problems occur partially due to the knowledge vaporization and lack of information about the design decisions, not only the ones taken and implicitly embedded in the architecture, but also alternative design decisions that were not implemented [59].

Architectural design decisions are defined in [59] as "*a description of the set of architectural additions, subtractions and modifications to the software architecture, the rationale, and the design rules, design constraints and additional requirements that (partially) realize one or more requirements on a given architecture.*"

This thesis represents each design alternative for a problem as the set of: description of the solution, design rules, design constraints, consequences, pros and cons. The analysis of the set of design alternative leads to a design decision.

2.3.2 Self-adaptation approaches

Adaptation refers to the ability of the system to automatically adjust their own behavior in response to changes in the operating environment [60]. Adaptation is seen as a key enabler for the development of autonomous systems and autonomic computing.

In 2003 Kephart, introduced the MAPE-K model with the IBM Autonomic Computing Initiative [61]. The MAPE- K model is a feedback loop and it stands for Monitoring, Analyzing, Planning and Executing over a Knowledge base. The MAPE-K loop forms the basis of self-adaptive systems feedback loops and has become the reference model in self-adaptation. The MAPE-K loop can be seen in the Figure 2.1.

Self-adaptive systems can be seen as an umbrella term to cover different areas involved in adaptation and are studied from different perspectives: software architecture, requirements engineering, middleware, component-based development, control systems theory among others [62]. These different approaches solve their specific domain problems with different architecture configurations.

Next, we briefly describe the different approaches used in self-adaptive systems. Several software architectures have been proposed for each approach. The architectures are studied in more details in Chapter 4.

2.3.2.1 Architecture-based adaptation

These systems are aware of its own architecture and structure it to modify the different relations between software elements, components and properties drive the adaptation. The adaptation logic utilizes a combination of the architectural models, metrics and policies [62].

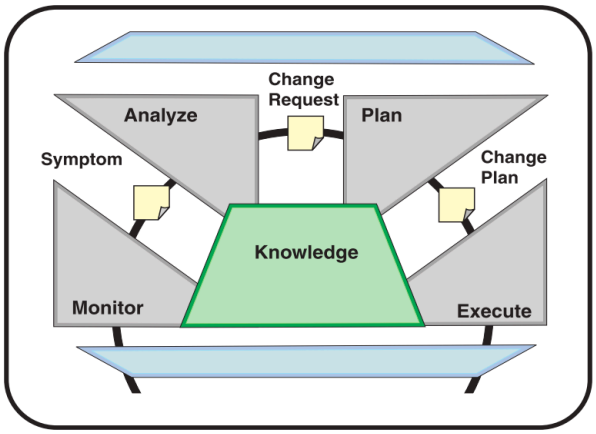

Figure 2.1 - The MAPE-K loop presented by Kephart [61]

2.3.2.2 Reflection-based adaptation

This approach uses the ability of the system to reflect in the behavior and the architecture level. The reflection can refer to the observation of the system's own behavior (introspection) or the reaction to of the introspection results (intercession) [62]. During the adaptation, behavior models of the system are used for analyzing the changes in the system.

2.3.2.3 Control-based adaptation

Control engineering theory is an extensive field that deals with the operation of dynamic systems and process. The objective of a control system is to provide control actions that stabilize a system in an optimal manner in respect to the desired metrics. Control theory already developed a number of different approaches to deal with unstable systems in discrete/continuous space, open/closed loop, linear/non-linear systems, deterministic/stochastic systems and centralize/decentralized systems. Several self-adaptive systems architectures are based on concepts develop by control theory and mimic many of the functions available in control systems. The MAPE-K loop itself can be seen as an extension of control approaches to include a shared knowledge repository [62].

2.3.2.4 Service-oriented adaptation

Service-oriented adaptation base the policies in using multiple small, encapsulated and autonomous services to enable different functionalities during the adaptation. During adaptation rules, policies, constraints and goals activate and exchange different services. These architectures build adaptation logic frameworks and define a communication layer for these services [62].

2.3.2.5 Agent-based adaptation

This approach uses multiple autonomous pieces of software that cooperates in a multi-agent system for a common task. The different agents share common goals, cooperate and communicate, but accomplish different tasks in decentralized and dynamic conditions. The main focus in the development of these type of systems is on how to plan each agent without a central coordination unit. Several agent-based approaches are inspired by nature systems, such as immune systems, firefly synchronization, chemotaxis among others [62].

2.3.2.6 Learning systems adaptation

This type of approach aims at continuously optimize the system structure, parameters or algorithms in regard to performance or cost metrics. This approach makes a higher usage of the shared knowledge system and learning approaches such as artificial intelligence, evolutionary programming, and reinforcement learning. These architectures can behave more actively to changes in the environment compared to the traditional reactive control-based approaches [62].

2.3.2.7 Requirements engineering for self-adaptive systems

Self-adaptive systems utilize different forms of requirements, such as system definition requirements, requirements at runtime for adaptation and adaptation mechanisms requirements. Proposes for requirements for self-adaptive systems include modeling uncertainty, relaxation of non-critical goals compared to emergency and high-level goals and runtime capabilities for requirements engineering.

Although self-adaptive systems can be studied several different perspectives, most self-adaptive systems approaches can be mapped into one or more MAPE-K loop parts [62]. This thesis reviews, in Chapter 4, several different architectures from each self-adaptation approach and analyzes how these architectures can serve as basis and leverage research in automated online experiments.

2.4 Conclusion

This chapter reviews the three main aspects discussed in this thesis, online controlled experiments, multi-armed bandits and software architectures for adaptation.

Online controlled experiments are relevant for all discussions presented in Chapters 4, 5, 6, 7 and 8. The experimentation techniques subsection gives an overview of the types of experiments and some of their limitations. These limitations often lead experiment organizations to look into alternative experimental designs such as multi-armed bandits designs. The subsection in the experimentation models is closely related to the discussion on the limitations and restrictions of experimentation in embedded systems, in Chapter 5, the experimentation process, Chapter 8, and the pitfalls in using multi-armed bandits in online experiments, Chapter 7.

Multi-armed bandits algorithms are used in online experiments and is one of the techniques used to automate the experiment execution. This discussion is relevant for Chapters 4, 6 and 7. We provide a brief overview of this class of algorithms and describe three widely used algorithms in more details. This overview is relevant to the discussion in Chapters 6 and 7. Although it not the focus of the chapter, the case study proposed in Chapter 4 also solves a multi-armed bandit problem with a domain specific heuristic.

The discussion on software architecture and adaption is relevant to Chapter 4 as it analyzes several architectures from different approaches to develop an architecture framework that supports automated online experiments.

3 Research Objective and Method

This thesis investigates the development and usage of automated experiments in software systems from the perspectives of the software architecture of an automated experimentation system, the usage of machine learning algorithms such as multi-armed bandits for automated experiments and the experimentation process. This chapter presents the goals, the research questions, the general research methods and the validity threats of this research. A detailed description of the research methods used in each study is given on the respective method section of chapters 4 to 8.

This chapter is organized as follows. First, this chapter provides a description of the research objective, the research goals and the research questions. Second, this chapter discusses the overall research methods. Third, this chapter provides an overall discussion of the threats to validity and strategies used to mitigate then. Finally, this chapter provides an overview of the key contributions of this thesis.

3.1 Research objective and research questions

The overall objective of the Ph.D. research is to *analyze how to automate different types of experiment and how companies can support and optimize their systems through automated experiments*. In order to achieve this overall research goal, this licentiate thesis divides this objective in four goals.

3.1.1 Goal 1 – To analyze how automated experiments can be supported from a software architecture perspective

Online controlled experiments have been discussed in a number of different publications, ranging from the experimentation process [3], [63], how to conduct experiments in online systems [6], [21], companies best practices [23], [39], [64], [65], statistical analysis [22], [66], [67] and machine learning algorithms [34], [68], [69]. Automated experiments research has focused more on algorithms and automating statistical analysis [28], [31], [33]. While there is a need to develop better algorithms to fully automate experiments, the architectural functional and quality requirements can have a significant impact on an automated experimentation system. Self-adaptive systems share several similar functional requirements with experimentation systems, in particular in optimization experiments. Self-adaptive systems research has developed a number of different architectures capable of allowing a system to self-optimize in procedures similar to running controlled experiments. Drawing inspiration and results from self-adaptive architectures, this chapter aims to understand how automated experiments can be supported from a software architecture perspective. This goal is represented in the research questions below:

RQ1: What are the architectural considerations when designing an automated experimentation system?

RQ1.a: What are the architecture qualities that support an automated experimentation system?

RQ1.b: What are the necessary architectural design decisions for an automated experimentation system?

This goal and these research questions are studied in publications A and B and are discussed in Chapter 4. This chapter discusses how to architect an automated experimentation system, providing architectural qualities and design decisions. It also develops an initial architectural

framework and instantiate it in an autonomous mobile robot to address the problem of human-robot proxemics distance.

3.1.2 Goal 2 – To identify how existing systems, including embedded systems, can use continuous experimentation to optimize software features

Online controlled experiments are mostly studied from the perspective of web-facing companies [6], [35], due to the simpler way to deploy new software to users, the real-time data collection, the reduced number of deployed versions and the high number of users. However, as embedded systems companies evolve their development practices and move towards continuous integration and deployment [44], they also see the opportunity of faster action and reaction to customer feedback and the opportunity to better understand their customers' needs [44] through continuous experiments. Understanding the challenges faced by embedded systems companies when adopting online experimentation practices and the strategies used to overcome these challenges is key when devising and implementing an experimentation system that can be used in a broader scope than web-facing companies. This goal is represented in the research questions below:

RQ2: How can the embedded systems industry adopt continuous experimentation in its development processes?

RQ2.a: What are the recognized challenges towards continuous experimentation faced by the embedded systems industry?

RQ2.b: What are the recommended strategies to facilitate the use of continuous experimentation in the embedded systems domain?

This goal and these research questions are studied in publication C and discussed in Chapter 5. Some of our industrial partners in the embedded systems domain see the potential of continuous experimentation in their development activities. However, they face different challenges compared to web-facing companies when adopting continuous experimentation. This chapter explores these challenges and how these industries are addressing these problems. The strategies indicate the need for novel techniques and automated experiments which might be a key technique for faster adoption of experiments in this domain.

3.1.3 Goal 3 – To analyze how different algorithms for automating experiments influence the outcome of an experiment

The majority of controlled experiment applications do not have models or representative datasets that can be used to drive inferences offline. Because of that, most experiment ideas must be tested online to observe customers' behavior and preferences. However, in some experiments, traditional methods such as A/B/n, full-factorial and fractional factorial experiments [36] perform poorly in terms of regret minimization and user allocation, particularly in the presence of poor variants. Multi-armed bandits algorithms [56] are a class of algorithms, under the reinforcement learning umbrella, that addresses some of these problems.

This class of algorithms is studied in the context of optimization experiments in automated online experiments as it simplifies the execution of experiments in the user allocation, the decision criteria and the minimization of regret. Within the scope of this goal, we investigate the usage of these algorithms in collaboration with industrial partners. This goal is represented by the following research questions:

RQ3: How can an automated experimentation system be integrated with existing systems to optimize feature parameters?

RQ4: What are the restrictions and pitfalls associated with multi-armed bandit algorithms in online experiments and how to overcome these restrictions and pitfalls?

This goal and these research questions are studied in publications D and E and they are discussed in Chapter 6 and Chapter 7. Chapter 6 discusses an improvement in a continuous-armed bandit algorithm for online experiments in the continuous space and describes how the instantiation of the automated experiment framework developed in Chapter 4 is used in an existing system in collaboration with Sony Mobile. The instantiation together with the proposed algorithm are used to automatically optimize system parameters.

Multi-armed bandits algorithms are often referred as a key technique to reduce costs and enable faster experimentation iterations. However, companies that have several years of experience and run thousands of experiments a year still avoid these algorithms in several cases, because of restrictions and pitfalls in the current implementations. Chapter 7 discusses the pitfalls, restrictions and strategies of using multi-armed bandits algorithms in online experiments.

3.1.4 Goal 4 – To identify what are the different parts of the experimentation process, the costs and how to automate the steps in the experimentation process

Chapters 3 to 7 discuss automation of experiments mostly in the execution step of the whole experimentation process. Although, the execution step influences the cost and outcome of experiments different parts of the process also take significant time and can influence the experiments' costs, effectiveness and trustworthiness. By understanding the different parts in more detail than what was presented in previous research we can move towards a more effective automated experimentation system for a software company. This goal is represented by the following research question:

RQ5: What are necessary steps and the set of activities in a trustworthy experimentation process?

This goal and the research question are studied in the publication F and it is discussed in Chapter 8. This chapter addresses partially the goal 4 by investigating the necessary steps to run trustworthy experiments in a case study with the Experiment and Analysis team at Microsoft. It serves as basis for the future work on how the different steps influence the cost of experiments, what steps would have a higher return on investment with automation, and how the automating experiments can achieve the same or higher level of trustworthiness in the experiment results as the current state-of-art practices achieve.

3.2 Overview of the research methods

This subsection describes the general research methods used in this thesis: literature review, case studies and the case companies that collaborated with us and empirical evaluations and simulations.

3.2.1 Literature review

Two literature reviews were used in this thesis as a primary research method intended to summarize existing evidence concerning technologies [70] and to identify gaps and limitations of existing practices or technologies. The research questions RQ1, RQ1.a and RQ1.b used a literature review to identify software architecture qualities for further development of automated experiments and to identify architectures that could be used as starting point or inspiration for an automated experimentation system. RQ2, RQ2.a and RQ2.b used a literature review in the definition and planning of the case study and as a source of triangulation. The scope, definition and results of the literature are presented in more details in Chapters 5 and 6.

3.2.2 Case study

Case study is a flexible empirical method aiming to investigate contemporary phenomena in their context [71], [72]. Case studies are conducted in real world settings and it has a trade-off between the level of realism and the level of control. Due to the degree of realism, case studies

are hard to study in isolation, therefore they do not create the same results in terms of causal relationships as controlled experiments, but they offer a deeper understanding of the phenomena under study [71]. In this sense, case studies are primarily used for exploratory and descriptive purposes with the aim of exploring relationships, generating new hypothesis and gaining insights.

This thesis follows the case study research guidelines presented in [71]. Our case studies are conducted in four stages, the definition and planning of the case study, the data selection and collection, the data analysis and the case study report. The definition and planning of the case studies were guided by reports of industrial practitioners and by identification of gaps and limitations of existing practices and technologies through a literature review. In the definition and planning of the case studies, we developed the research questions and interview protocols, which were revised by the involved researchers before the interviews. All conducted interviewees were selected based on their role, expertise and industry domain in connection with the research questions. The data collection in our case studies was primarily through semi-structured interviews with industrial practitioners in the fields of software engineering and data science, and by interview notes, shared documents, diagrams and exchanged emails. The data was analyzed by thematic coding [73], in which the interpretation of recurring codes consists of the main reported results. The case studies were reported in the listed publications of this thesis and in Chapters 4 to 8.

The research questions RQ2, RQ2.a, RQ2.b, RQ4 and RQ5 are investigated using a case study method. RQ2, RQ2.a and RQ2.b are exploratory questions aimed at understanding what are the current challenges in adopting continuous experimentation practices faced by the embedded companies and how these challenges are addressed. These questions are primarily exploratory and hard to discuss without considering the context of each company. Similarly, RQ4 is primarily exploratory and aims to understand the challenges associated with multi-armed bandits outside the controlled environment provided in previous research. RQ5 is also exploratory and it is aimed at identifying how a company, in its context, established a trustworthy experimentation process.

3.2.2.1 Case companies

The case studies were conducted with multiple companies. These companies were selected based on their domain, expertise in online experiments and multi-armed bandits. With the exception of Sony Mobile (Chapter 5) and the Microsoft Corporation (Chapter 8), the other companies wished to remain anonymous, as the studies often describe technology limitations, errors and pitfalls and limitations in current development practices. Below, we provide a brief description of the companies presented in Chapters 5, 6, 7 and 8.

Sony Mobile Communications is a subsidiary of Sony Corporation and is a leading global innovator in information technology products for both consumer and professional markets. One of the Sony Mobile's products is transitioning to data-driven development and aims to run experiments continuously throughout its development process. The product consists of a business to business solution, where the user of the software consists of employees of the company that requested the solution.

Microsoft Corporation is a multinational technology company that develops, manufactures, licenses, supports and sells computer software, personal computers, consumer electronics and services. Microsoft is one of the leading companies in online experiments running over 20 000 experiments a year [38] in multiple types of systems such as web, personal computer, mobile, embedded systems and cloud infrastructure.

Company I is a travel fare aggregator and travel engine provider. It develops booking and travel solutions used by both individuals and the travel industry. A/B testing methodologies are an integral part of the development process of the company.

Company II is a multinational company that provides telecommunication and networking systems. The company is adopting continuous development practices and is looking for new strategies to deliver more value to their customers by optimizing their products.

Company III is a global automotive manufacturer and supplier of transport solutions. As the company's products are continuously growing in complexity and software size, the company is looking for strategies to prioritize their R&D effort and deliver more value to their customers. As some employees have experience in web and pure software-systems development, experimentation is getting attention in some development teams.

Company IV is a global software company that develops and provides embedded systems software solutions related to autonomous driving technology for the automotive industry. Autonomous driving is an emerging and fast-moving technology and the company is looking to deliver competitive solutions faster by adopting continuous development practices.

Company V is a global software company that develops both software and hardware solutions for home consumers. The company already has experience running continuous experimentation in their web systems and is starting to run experiments in their hardware solutions.

Company VI is a multinational company that manufactures embedded systems and consumer electronics. In recent years, the company started to adopt experimentation in their software solutions and is looking for data-driven strategies in their embedded systems products. The interviewees were developers, managers and data analysts.

Company VII is a company that develops experimentation solutions for its customers. The company offers A/B/n, MVT and other experimentation tools for websites along with frameworks for experimentation in mobile platforms. The company developed their own statistics engine and offers solutions using MAB algorithms to customers. The company's customers include several multinational companies from different domains, from software companies, to entertainment, to large news agencies.

Company VIII is a software company focused on website optimization and offering experimentation tools and solutions for A/B testing and MABs. The company's customers include several multinational companies in North America, Asia and Europe.

3.2.3 Empirical simulations and evaluations

Simulations allow researchers to isolate the subject of study in a well-controlled and reproductible environment, to investigate its performance and correctness and comparison between alternative implementations [74]. In this thesis we conducted a number of simulations to investigate the behavior, performance and correctness of the developed systems and algorithms and to triangulate data reported in case studies.

In RQ1, RQ1.a and RQ1.b, the developed architecture was instantiated and evaluated in different simulated scenarios prior to its implementation in real scenarios as described in Chapter 4 and 6. The RQ3 is investigated with both empirical simulations and evaluations of the developed architecture framework from Chapter 4 in collaboration with an industrial partner. The RQ4 is primarily studied using a case study method. However, simulations where also used to triangulate the restrictions and pitfalls identified in the case study.

3.3 Threats to Validity

This section discusses threats to validity regarding how our research questions were answered.

3.3.1 Construct validity

Construct validity refers to what extent the operational measures represent what is under investigation according to the research questions. For example, if the interview questions are not interpreted the same way by the research and the interviewees [71].

To mitigate this threat to validity in the conducted case studies, we selected and interviewed subjects that had experience and worked with online experiments before, either in their current company or in previous experiences. Prior to the interviews, we designed the semi-structured interview guided by terms often seen in gray literature, in the company's technical reports and blogs. During the interview, we explained the new concepts that we were trying to investigate, we asked the interviewees to elaborate more on an ambiguous or off-topic answer and we rephrased questions when the interviewee did not understand what we meant. During the data analysis, if interviewees' comments where not clear, or if there was ambiguity or if the comments did not reflect the question, we contacted the interviewees to solve these problems.

3.3.2 Internal validity

Internal validity refers to whether the external factors could impact the results of the investigated factors, when causal relations are examined [71].

We recognize that this threat could potentially affect the results presented in this thesis. The results and strategies associated with RQ2.b, RQ4 and RQ5 were developed by the companies in their context. As the researcher only had access to the final result and descriptions of the strategies, it is not possible to investigate if other factors were more influential to the final result than the proposed strategies.

3.3.3 External validity

External validity refers to what extent the results of the study can be generalized to other situations outside of the investigated case [71].To mitigate this threat, we approached the investigation of the research questions in different ways. The architecture framework designed in RQ1, RQ1.a, RQ1.b and applied in RQ3 was based on results from a literature review and tested in different descriptive scenarios and case studies. It was instantiated in an academic mobile robot (Chapter 4), and in a larger mobile application (Chapter 6). Together with two industrial partners we are currently using this architecture in a broader scope in terms of application (mobile application and the related cloud backend infrastructure and in a telecommunication embedded system), algorithms (A/B/n, MVT, MAB, combinatorial bandits and continuous-armed bandit) and in different parts development (pre-deployment offline optimization, hardware in the loop optimization and post-deployment online optimization). Although using this architecture in different scenarios does not guarantee the generalization to other cases, it increases our confidence that it can be used in a larger scope than what we are testing.

For RQ2, RQ2.a and RQ2.b we tried to generalize the findings by selecting companies developing different embedded systems. We selected only challenges that could be triangulated with more than one company or that were already identified in previous research in online experiments. For RQ4, we contacted companies that have different levels of maturity in online controlled experiments, in different domains, and that develop, use and sell online controlled experiment systems with multi-armed bandits algorithms. Part of the empirical data was also triangulated with simulations to extend validity of the claims. For RQ5, although we conducted our work with only once case company, our empirical data was collected from experiences of several different products.

Although we tried to mitigate the external validity threat in different ways, without replication of these studies is not possible to confirm or extend the generalizability of the results.

3.3.4 Reliability

Reliability is concerned to what extent the data, the data analysis and the results are dependent on the specific researchers [71].

To mitigate this threat, we triangulated the data of all interviews and case studies with other sources, such as with previous research, replicating the results in simulation, comparing with reports from multiple companies and interviewees. In the development of the architecture and its applications, the source code is also available online.

3.4 Key contributions

The key contributions of this thesis to the development and understanding of automated experiments in relation to the research goals are:

Goal 1 – To analyze how automated experiments can be supported from a software architecture perspective

- Identification of software qualities to support automated experimentation.
- Description of a set of architectural design decisions for an automated experimentation system.
- Proposal of an initial architecture framework for automated experimentation that was evaluated in an autonomous mobile robot.
- Identification of key research challenges that need to be addressed to further support the development of automated experimentation.

Goal 2 – To identify how existing systems, including embedded systems, can use continuous experimentation to optimize software features

- Identification of the key challenges faced by embedded systems companies when adopting continuous experimentation and proposal of a set of strategies and guidelines to overcome these challenges.

Goal 3 – To analyze how different algorithms for automating experiments influence the outcome of an experiment

- Development of a bandit algorithm for continuous space parameters (LG-HOO) that considers online experimentation constraints and statistical evidence that supports the usage of the algorithm in online experimentation scenarios.
- Validation of the LG-HOO algorithm in an industrial case study.
- Identification and analysis of restrictions and pitfalls of multi-armed bandit algorithms applied to online experiments.
- Identification of strategies for practitioners to avoid pitfalls in online controlled experiments.
- A set of guidelines for practitioners to select an experimentation technique that minimizes the occurrence of the identified pitfalls.

Goal 4 – To identify what are the different parts of the experimentation process, the costs and how to automate the steps in the experimentation process

- Identification of the differences between existing experimentation models and the current state-of-the art.

- The proposal of a new experimentation process framework consisting of two models, an activity model and an experiment metric model. This framework captures and explain the experimentation process in a level of detail not provided by any of the experimentation models discussed in previous research.

3.5 Conclusion

This chapter discussed the research objective of this Ph.D. This objective is divided in four main goals. For each goal, we proposed a set of research questions. The research methods are discussed in general and how they were used to address each research question. We discuss the four main threats to validity and how we tried to mitigate them. We conclude this chapter with an overview of the key contributions for each of the proposed goals.

4 An Architecture Framework for Automated Experimentation

This chapter is based on the following publications:

Your System Gets Better Every Day You Use It: Towards Automated Continuous Experimentation
David Issa Mattos, Jan Bosch and Helena Holmström Olsson
2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Vienna, 2017, pp. 256-265.

More for Less: Automated Experimentation in Software-Intensive Systems
David Issa Mattos, Jan Bosch and Helena Holmström Olsson
In: Felderer M., Méndez Fernández D., Turhan B., Kalinowski M., Sarro F., Winkler D. (eds) Product-Focused Software Process Improvement. PROFES 2017. Lecture Notes in Computer Science, vol 10611. Springer, Cham

Chapter Summary
Innovation and optimization in software systems can occur from pre-development to post-deployment stages. Companies are increasingly reporting the use of experiments with customers in their systems in the post-deployment stage. Experiments with customers and users are can lead to a significant learning and return-on-investment. Experiments are used for both validation of manual hypothesis testing and feature optimization, linked to business goals. Automated experimentation refers to having the system controlling and running the experiments, opposed to having the R&D organization in control. Currently, there are no systematic approaches that combine manual hypothesis validation and optimization in automated experiments.

This chapter focuses on how architectural aspects of adaptation can contribute to support automated experimentation. Based on a literature review and architectural design decisions, we developed an initial architecture framework to support automated experimentation. This architecture framework is instantiated in a case study using an autonomous system.

The contributions of this chapter are threefold. First, it identifies software architecture qualities to support automated experimentation. Second, it develops an initial architecture framework that supports automated experiments and evaluates the framework with an autonomous mobile robot in a human-robot proxemics distance problem. The architecture is developed through a set of design decisions evaluating pros and cons on how different architectures impact the instantiation of the system Third, it identifies key research challenges that need to be addressed to support further development of automated experimentation.

4.1 Introduction

Autonomous systems already play a big part in several areas, from financial markets, industrial robots, airplane navigation systems to the development of autonomous vehicles. These systems are deployed in uncertain environments and contexts with an ever-increasing demand to work more autonomously. As these systems become more and more complex, it is not clear how the

developed features could be contributing to the system and whether they are delivering the expected value [27].

Previous studies show that in the development of systems, traditionally, the prioritization of a feature is usually driven by earlier experiences and beliefs of the people involved in the selection process [9]. The development of the new features should be, ideally, data-driven and done systematically [46]. The development of a full feature from conception to user deployment can result in inefficiency and opportunity cost if it does not have a confirmed value for the customer. Early customer feedback is important to determine and validate the feature value [11]. If a gap between the expected and the actual value of a feature is identified, the feature under development can be refined or abandoned. Additionally, companies continuously collect data from deployed software [18], [19]. Although not widely used in industry, post-deployment data can be used for both continuous optimization and improvements and to drive innovation in features of the existing products [19]. Testing and experimenting with customers and users are used as problem-solving processes and are critical to organizational innovation [10].

Frequently, complex systems and user behavior in development of software-intensive systems lead to opinion-based decisions captured in traditional requirements [6]. This mismatch between user behavior and opinion-based requirements is leading industry to change from requirements-based to experiment-based development [10]. Systematic experiments provide a level of understanding about what really works and leaving opinion-based decisions behind [75]. Web-facing companies report the use of experiments with customers and that these experiments can lead to a significant learning and return-on-investment [6]. Several companies report the use of A/B experiments for confirming feature value through hypothesis testing and for feature optimization [6], [26]. In [10] online experiments are identified as a technique to drive innovation during development and post-deployment of a system.

However, traditional manual controlled experiments can be an expensive way to optimize a system. Validation runs online and can last for several months [6], the metrics used by different teams can lead to conflicting business goals, and it can be hard to reason on the system when dealing with hundreds of different observable variables [76]. In this scenario machine learning and artificial intelligence techniques can aid the research and development team to run optimization procedures to existing features more efficiently.

In customization and recommender systems there is an increase use of machine learning (ML) and data-driven approaches. Techniques such as deep learning, reinforcement learning (RL), deep reinforcement learning, multi-armed bandits are getting large attention as companies (Google, IBM, Microsoft, Spotify, Facebook among others) successfully report the use of those techniques in their systems. Machine learning communities continuously report solutions to a vast range of difficult problems. However, ML also raises several software engineering problems, such as testing, system validation, supporting infrastructure and increase of technical debt in the system [77]. Many ML algorithms have mathematical proofs but with the ever-changing landscape that those algorithms interact, system validation becomes a hard problem. Controlled experiments are used to evaluate both the ML value and system behavior [78].

As companies start to build autonomous systems with an increasing level of autonomy, experiments are seen as scientific way to learn and adapt the system behavior. The uncertainty raised by the environment, the interaction with humans and other agents all impact in the system behavior in unknown ways. It is unfeasible to grow the size of the R&D team with the increasing demand for experiments. This calls for the use automated experiments, where the R&D team can build part of the functionality and guardrails where the system can experiment and learn from this process continuously and autonomously [27].

As companies are moving towards experiment-based development, they face several challenges such as: experiments in live field are time and cost expensive, experiments from different teams can lead to conflicting goals, and the lack of a systematic approach to run experiments in different domains leads to several experiment platforms. [6], [23], [27], [76]. To address some of these challenges the communities studying experiments in software can be viewed as moving towards automated experiments. Important research deals with automating data collection and data analysis [76], facilitating the deployment of new experiments in web systems [21], dealing with overlapping experiments [23], leveraging experiments with log data [34] and automated experiments from an algorithm perspective [32].

Automating tasks and allowing systems to perform adaptation online play a role part in the development of autonomous systems. Autonomous and adaptive systems are able to automatically adjust their behavior at runtime in response to changes in the operating environment [79]. Over the past fifteen years, different domains such as the self-adaptive systems community studied and developed several software engineering approaches and techniques for adaptation. While the ML is mostly concerned with algorithms and the theoretical basis of learning, these communities study other software engineering challenges.

Running continuously automated experiments in the already deployed systems, with both the aim of improving the system behavior and confirming the delivered value, require novel software architectures and engineering approaches [27]. Techniques from these different perspectives can serve as a basis for bridging from manual hypothesis experimentation and experimentation for optimization to automated experiments.

Although we recognize the important role ML and other artificial intelligence techniques have in the development of automated experiments, this chapter focuses on architectural aspects to support different techniques in automated experiments for both validation as well as optimization of features. Continuous optimization through automated experiments leads to systems that get better every day we use them.

4.2 Research Method

The research method used in this chapter is divided in three phases.

4.2.1 Phase I

In the first phase, we reviewed literature to identify relevant software architectures for experimentation and adaptation and how these architectures solve domain-specific problems.

For the literature search, we looked for software architectures in the different domains and applications related to experimentation:

```
(TITLE-ABS-KEY("software architecture") OR TITLE-ABS-KEY( "software framework")) AND
(TITLE-ABS-KEY("experimentation") OR TITLE-ABS-KEY("A/B tests") OR TITLE-ABS-
KEY("split tests") OR TITLE-ABS-KEY("self-adaptive systems") OR TITLE-ABS-
KEY("controlled experiments"))
```

This query was used in the indexing libraries SCOPUS and Science Direct, that covers the large research libraries. As a result of the literature search, we identified 410 papers with relevance to this search. From these papers, we identified 178 based on reading the abstract and identifying the relevance in relation to our research topic. From this we selected 34 papers that describe relevant concepts and software architectures for experimentation, adaptation and controlled experiments. In addition to the literature search based on the identified key search terms, through cross-reference we identified a set of additional 18 papers additional papers with relevance for our research.

The analysis of the selected works provided us two main outcomes: (1) the identification of relevant architectures for experimentation, adaptation and controlled experiments and (2) identification of software architecture qualities that can support automated experimentation. The qualities are described in detail in Section 4.3.

4.2.2 Phase II

Based on the outcomes of phase I, we scoped the second phase to understand how each identified architecture implements the software qualities. In this phase, we revisited the identified architectures to identify how they implement or solve a problem related to each of the qualities. Our analysis, described in Section 4.4 indicated that none of the architectures could be used as is in the automated experimentation problem but they can be used to derive a new framework that could support automated experimentation.

4.2.3 Phase III

Based on the analysis performed in phase II, phase III focused on developing an initial framework that could support automated experiments. This framework is accompanied by a set of design decision to facilitate its modification and instantiation to be used in different domains. An initial version of this framework was instantiated in a proxemics distance problem in human-robot interaction. This human-robot interaction problem is currently being studied using manual experiments and therefore is a good candidate to try the developed automated experimentation framework.

4.3 Architecture Qualities

This section discusses the identified architectural qualities that support the idea of automated experimentation.

The identified qualities are outcomes of phase I of the research method, described in Section 4.2.1. In our research, we identified six qualities that would constitute our approach towards automated experimentation. However, this list of qualities is not static. Further research on the area might extend this list to include different qualities. Additionally, the identified qualities are seen as desired qualities rather than required qualities. Excluding one or more of the qualities might be needed to implement a domain specific setup for automated experiments. Therefore, this list is ordered in terms of importance.

4.3.1 External adaptation control

Adaptation can be divided in two types of adaptation control: internal and external [80]. Internal adaptation refers to interlace application logic with adaptation logic. External adaptation refers to the use of an adaptation manager that coordinate adaptation with the use of sensors and effectors in the managed system (application logic). The use of an external manager increases maintainability of the system. However, it introduces an overhead to the system, penalizing performance. Traditional controlled experiments are analyzed offline. All the data is extracted, analyzed to support a decision. Depending on the decision the system is manually modified to incorporate this decision. Systems running multi-armed bandits and other ML algorithms are usually incorporated in the feature application code. The external adaptation control was identified as an important quality for automated experiments because it allows separating the application logic from the experiment logic. This helps both automated experiments in controlled settings and in optimization settings, by facilitating to add automated experiments to existing features and removing it from features that already reached the static system loop [27].

4.3.2 Data collection as an integral part of the architecture

Software-intensive systems can gather large amounts of data in real-time, from the context, from the internal system as well as from users of the systems. Collecting data for online and

offline analysis by both the system and the R&D should be an integral part of the architecture. An emphasis is given in this step as the development of software should be data-driven [9].

As most adaptive systems are based on the MAPE-K reference framework, data collection is usually seen as part of the monitor phase on the MAPE-K loop. The managed system exposes several observable internal states through the sensors touchpoints [81]. Data is collected continuously and filtered and later analyzed by the Analyze component to detect changes and decide whether adaptation is needed or not.

Machine learning systems are easy to develop but hard to maintain [77] due to hidden technical debt. One of the causes is the high diversity data and unstable data dependencies that rises by the extensive use of glue code. Strategies to create common API's allow a more reusable infrastructure supporting integral data collection and the external adaptation.

4.3.3 Performance reflection

As part of the learning process, the system should have performance reflection as an important part of the architecture. Performance reflection consists of evaluating the current behavior according to an expected value function. The experimental methods also use the performance reflection to validate the observed behavior.

In the conceptual solution presented in [27], evidence-based engineering refers to validate the deployed feature based on experiments and the value it delivers. Also, it was identified three levels of evidence-driven development feedback loop: the R&D loop, the dynamic system loop and the static system loop. Performance reflection occurs in the dynamic system loop in which the system can compare its delivered value with the initially expected value.

If the system does not keep track of its adaptation performance it is not possible to have an evidence that the experimentation and the learning process is increasing value to the system.

4.3.4 Explicit representation of the learning component

Most of the current approaches in self-adaptive systems recognize the importance of learning, but few of them support an online learning process [62]. Most architectures use predefined and reactive adaptation plans.

The field of ML provides several algorithms and techniques tailored to domain problems. Experimenting features in different domains requires modification of the learning algorithm. An explicit representation of the learning component facilitates introducing and maintaining (mitigating entanglement effects) learning algorithms in the optimization process. We believe that an explicit representation of the learning component is necessary because it reinforces the learning characteristic of the systems and facilitate the reuse of the learning component and learning algorithms in other features.

4.3.5 Decentralized adaptation

Centralized control refers to using a single adaptation manager to control the adaptation. Decentralization refers to each adaptation where each sub-system has a full adaptation manager. The use of decentralized adaptation control improves the performance of the system, splitting responsibility between sub-systems. Hybrid approaches might use different patterns, such as the IBM hierarchical structures to allow adaptation [82].

Different systems running automated experiments will require different levels of decentralization. Centralized systems can grow quickly in complexity handling several features under experimentation. Traditional A/B experiments rely on centralized coordination. However, as the infrastructure grows decentralized patterns start to emerge. Tang et al. [23] describe a hybrid approach, dividing the system in layers with different levels of overlapping

crossing multiple domains. We believe that automated experiments will decentralized solutions as the number of experiments scale.

4.3.6 Knowledge exchange

Knowledge exchange can help systems to share learned and experimented solutions [81]. Collaborative learning is an increasing topic of interest in ML and in experimentation. Systems instantiated or users experimenting might not be completely independent or randomized. Algorithms for solving this sort of problem are studied within networked A/B tests [76] and counterfactual reasoning [34]. As the feature being experimented might also be evaluated by the development team in terms of the value it gives, the sharing component should also allow communication with the development team. Knowledge exchange can be seen in the work by Fisch et al. [83] and in the area of collaborative feedback and Collaborative Reinforcement Learning [81].

4.4 Architecture Analysis

This section discusses briefly the analyzed architectures and how they can contribute to the development of automated experiments. We analyzed the existing architectures for self-adaptive systems in relation to the identified qualities as stated in the research method.

Architectures from different domains were selected in order to minimize the bias in selecting only a few. However, this list does not aim to be complete in respect of all existing architectures for adaptive systems. A description of the different approaches can be found in [62]. **Error! R eference source not found.** provide an overview of architectures analyzed in this work.

Table 4.1 – Approaches and architectures for self-adaptive systems

Approach	Architecture
Architecture-based	Rainbow framework [84], 3-layered approach [85], Archstudio [86]
Reflection-based	DynamicTAO [87], CARISMA [88]
Control-based Architectures	Model Predictive Control, MIAC, MRCA, Gain scheduling, Cascaded control [89]
Service oriented	SASSY [90], MetaSelf [91], MOSES [92], MUSIC [93]
Agent-based	CRL [81], Unity [94], MOCAS [95]
Learning systems	FUSION [96], Controller-Observer [97]
Requirement Engineering methods	LoREM [98], Zanshin [99]

The architectures were analyzed, classified and ordered according to the six qualities listed in Section 4.3. Figure 4.1 shows the obtained classification of the analyzed architectures according to the listed qualities. Each of the six qualities is connected to the architectures that fulfill these qualities.

The summary table provides an overview of how the different architectures relate to the desired qualities. However, it is possible to see that most of the approaches deal with only a few of the identified qualities. The architecture that satisfy most of the qualities is the FUSION framework [96] and the developed architecture framework in Section 4.6 is inspired by this framework.

The FUSION framework can be seen as a variation of the MAPE-K loop for architecture optimization. It uses a learning approach to drive adaptation of features. It is focused in

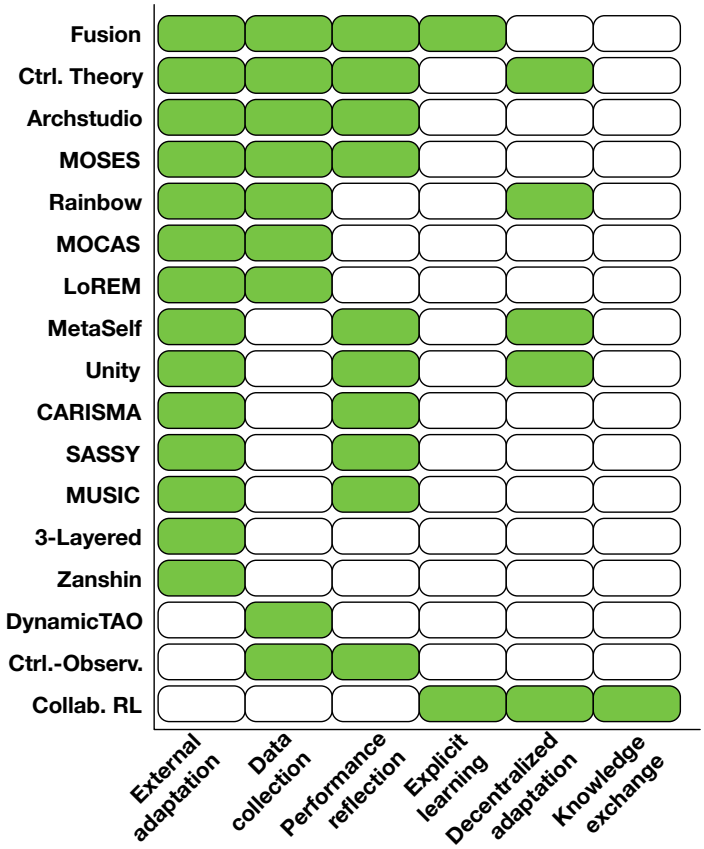

Figure 4.1 - Summary of the classification of the architectures with respect to the architecture qualities. In green, are the qualities satisfied by the architecture. The architectures were ordered according to the relative importance of the quality.

optimization in the architectural level and it is independent of the learning algorithm. The FUSION framework uses a centralized external approach. The adaptation manager is divided into two cycles, the adaptation and the learning cycle. The learning process is based on measurements of the system. After the collection, the learning cycle identifies any emerging pattern and refine the induced model in a knowledge base. Then, the knowledge base is used in the adaptation decision. Objective functions are used as metrics for the learning process and for the decision making.

This framework brings together many of the identified qualities, except for the decentralized adaptation and the knowledge exchange. However, it focuses on optimizing from learning from existing features in the knowledge base in the architecture level. Automated experimentation is focused on learning and optimizing business goals to drive post-deployment innovation.

4.5 Architecture framework design decisions

The introduction and background sections provided the motivation for the development of this architecture framework. We assume the system is already developed using a particular set of technologies and following a specific software architecture. The automated experimentation architecture framework will enhance the system capabilities using the existing system infrastructure. The system under experimentation (SuE) refers to the part of the system that is being experimented. For simplification, we divide a system into three levels: system, subsystems and components. Although we envision to automated experimentation in different levels of a system, we started the development of an automated experimentation architecture framework in the component level only.

4.5.1 Functional requirements

Next, we describe the functional requirements of the experimentation system. These requirements were obtained by analyzing the automated experimentation problem, the descriptive scenario presented in [27] and the attributes needed to support it in a family of systems.

- The experimentation system allows the system under experimentation to run experiments, measure its own behavior and learn from this process. Based on predefined metrics the system under experimentation will improve its behavior aided by the experimentation system
- The architecture framework should support different learning algorithms and not be restricted to one in particular.
- If the system under experimentation is part of a family of systems It should be able to learn and share learned solutions with the other systems.
- More than one feature can be experimenting at the same time. Confounding factors associated with multivariable experiments should be considered.
- The system should support manually predefined variations, as well as an automatically generator of new variations.
- The system should keep track of some guardrails metrics [27] while experimenting. If the system is not in the experiment boundaries, a predefined safe version should be active.

Together with the architecture qualities, the functional requirements will serve as basis for the development of the architecture through the set of design decisions below.

4.5.2 Problems, potential solutions and decision

We describe the design decisions using a set of {problem, potential solutions, decision}. Some of these problems are recognized from research literature or industry challenges in both the controlled experimentation, software architecture and adaptive systems.

The architecture design decisions was proposed in [59] and is considered as a first-class representation in the architecture representation. This representation allows decisions to be reevaluated as modifications are needed for the system to evolve or adapt to a new domain.

4.5.2.1 Type of experimentation

Problem: the experimentation can be integrated into the application logic or developed as a separate part of the application logic.

Motivation: several systems that can benefit from experimentation are built using different architectures, frameworks and technologies.

Potential solutions

External experimentation manager.

Description: this solution creates an external manager to the system. This separates the experimentation logic from the application logic.

Design rules: the experimentation manager interacts with the system through its interfaces. The system should continue to work even if the experimentation manager is removed.

Design constraints: the manager should only interact with the system through its interfaces.

Consequences: the becomes loosely coupled with the system.

Pros: maintainability of both the system and the experiment.

Cons: reduced timing performance of the system. Additional complexity as it introduces a communication layer and new interfaces.

Internal to the application logic.

Description: this solution incorporates the experiment into the application logic, combining into one system.

Design rules: the experiment design is incorporated into the development workflow of the systems.

Design constraints: the feature developments should handle the experiment design.

Consequences: the correct functionality of the system becomes dependent on the experiment logic. Difficult to reuse code between similar experiments.

Pros: integrated with the current development workflow. Better timing performance compared to an external manager.

Cons: reduced maintainability. If there is a need to change the experiment/learning algorithms or the application functionality, affects both the application and experiment logic.

Decision: The decision is made to use the external experimentation manager solution, which decreases maintainability, facilitates changes and new algorithms and increases the reliability of the system. This decision is also acknowledged in the manually controlled experimentation [21] and in adaptation architectures .

4.5.2.2 The degree of decentralization.

Problem: the external experimentation manager can be integrated with the systems in different degrees of decentralization.

Motivation: the degree of decentralization of the system impacts both performance, maintainability and scalability.

Potential solutions

Centralized experimentation manager.

Description: in this solution, the system has only one manager coordinating all experiments.

Design rules: the experimentation manager is responsible for getting all the system data and coordinating the experiment at all levels, from sub-systems and component to feature-level.

Design constraints: only one instantiation of the manager. the manager should be able to control independent experiments and different algorithms.

Consequences: each experiment is controlled in only one central place.

Pros: only one place to keep track of the experiments. Easier to coordinate different experiments and avoid confounding factors.

Cons: a single point of failure, experiments not related to each other can be influenced by the manager performance. Scalability might be affected when dealing with several experiments at the same time.

Decentralized experimentation at feature-level.

Description: in this solution, several experimentation managers are deployed in feature-level.

Design rules: each manager is responsible only for the feature it is interacting with.

Design constraints: a feature experiment should not alter other features experiments. The managers should not share resources with each other.

Consequences: several instances of experiment managers are deployed.

Pros: provides a way to scale the system and a failure in one manager does not affect the other experiments. Customization of the manager for a particular experiment does not imply changes in a larger manager.

Cons: different software versions of a manager might be running at the same time, reducing maintainability. In this approach, lack of synchronization between the managers can introduce confounding factors between the experiments.

Decision: The decision is made to use a decentralized approach. Data-driven companies experimenting at large scale report using decentralized tools to run experiments, segmenting the system to allow concurrent experiments [21], [23]. Although, in simple cases a single manager might be easier, in systems where the software is distributed in different computational nodes (such as robotics, automotive and other embedded systems) a single manager is not an alternative. This approach allows to experiment at different levels of the system hierarchy. Moreover, parts of the system can be isolated (e.g. security) from each other and a single manager in those parts might break other architectural rules of the application logic.

4.5.2.3 Confounding factors.

Problem: different experiments can influence the same aspect of the system behavior. In this scenario, it is hard to correctly interpret which experiment lead to the correct change (confounding factors).

Motivation: in a decentralized architecture, lack of synchronization on the on-going experiments can lead to confounding factors.

Potential solutions

Centralized experiment coordinator.

Description: the system has only one experiment coordinator to manage all the experiments. The other parts of the architecture would remain decentralized.

Design rules: all the decentralized components communicate and coordinate the experiments with a central coordinator.

Design constraints: each decentralized component does not have full autonomy to run the experiment. They should receive permission from the central coordinator.

Consequences: every new experiment should register and communicate with the central coordinator.

Pros: simplify the coordination process in a single instance.

Cons: reduces the reliability of the system by having a single point of failure. Changes in the experiment coordinator can affect all experiments. Changing the

coordinator with running experiments can insert bias in the sample and possibly affect all running experiments (in the order of hundreds in large data-driven companies [23]).

Decentralized conflict manager.

Description: each experiment instantiation has its own experiment coordinator and a conflict manager. The conflict manager is responsible for keeping track of the current experiments in the system and the experiment coordinator is independent of other experiments.

Design rules: the conflict manager signals when starting an experiment and it is notified when other experiments start or stop. This allows the experimenting feature to keep track of potential confounding factors.

Design constraints: experiments should be independent of each other and communication happens decentralized.

Consequences: each experiment system is contained, not depending on the other experiments.

Pros: independent experiments reduce the risk of one experiment failure increasing the risk of failures in all experiments, facilitates the scalability of the number of concurrent experiments.

Cons: introduces an extra layer of complexity when running the experiments or create decentralized communication infrastructure between features.

Decision: The decision is made to use a decentralized conflict manager. Although this solution introduces an extra layer of complexity, it is the solution used in open source tools from companies that experiment in large scale [21], [23]. The extra complexity comes in the designing of the experiment inside the organization. Experiments in the system can be divided between different teams in a way that there is no overlapping. Google divides different experiments in a layer model [23]. Facebook uses namespaces mapped to independent segments of users [21]. Sometimes experiments go through several iterations until the result can be considered valid and the cost to affect several on-going experiments might be prohibited for an organization.

4.5.2.4 Information exchange.

Problem: The learning behavior can take a long time to converge in a system isolated. Moreover, the learning process might converge to a specific context where the system is inserted. This situation can reduce the value that the system is delivering.

Motivation: In the case of several systems being deployed, one system can help the learning process of the other systems. A system initializing with more advanced versions can converge quicker to a solution or initialize with a pre-defined solution.

Potential solutions

Central server.

Description: in this solution, the system sends experiment data back to the development team and to other systems through a central server. This data can be used to initialize new learning processes in other systems and be integrated into future developments by the R&D team.

Design rules: the system needs an infrastructure to communicate with the other systems and the R&D when it learns. A central server infrastructure is necessary to coordinate the information flow.

Design constraints: the system should have access to a reliable and secure network. Communication directly between systems is not allowed. Control over quality and origin of the information is transferred to the central server.

Consequences: The system is constantly sharing information with a central server. This solution requires a backend infrastructure.

Pros: the company behind the system has control of the information flow and the information quality.

Cons: a central server introduces a point of failure in the system. If the server is down communication between systems is compromised.

Peer-to-peer.

Description: this solution equips the system the ability to handle communication directly between systems.

Design rules: each system should be able to discover other systems and manage the communication and information flow.

Design constraints: control over the quality and origin of the information is constrained to each system. Consequences: this solution can use the same network infrastructure as the central server solution. However, each system should be able to discover other systems, authenticate and verify the quality of information.

Pros: reduces the dependency on a central server, eliminating a possible point of failure. This also facilitates communication directly between the systems.

Cons: this solution introduces an overhead to verify the source and quality of information in each system. Systems producing incomplete or faulty information become harder to detect. The research and development team does not have access to the full information shared between systems.

Decision: The decision is made to use a central server to coordinate information exchange. Knowledge, tools and solutions for running central servers and handling communication are largely available. This solution also provides a way to control the information flow concerning user's private usage data, quality and origin of the information. Moreover, it allows sharing data with the R&D team.

4.5.2.5 Guardrails.

Problem: experiments should happen in predefined boundaries and conditions. Running experiments outside the experiments boundaries can decrease the performance, deteriorate systems metrics or business goals, and generate invalid experiment results.

Motivation: systems experimenting outside their experiment boundaries can lead to situations that put the user or the system safety in risk, or deteriorate metrics that were not considered when designing the experiment

Potential solutions

Restrict when the system can experiment.

Description: in this solution, the R&D team only allows experiments in systems that fulfill the experiment criteria in advance.

Design rules: the experimentation framework is not responsible for selecting when the system can experiment. This is done by the R&D team.

Design constraints: the system does not keep track of the boundaries or the experiment context.

Consequences: if the system or the user change context, an experiment might be running outside boundaries. Unless the R&D teams take action, the system will continue running the experiment.

Pros: this solution is easier to implement when experimenting a low number of features in a small user base. The system does not need to monitor the context of the experiment.

Cons: this solution can introduce bias in the experiments, because of the population selection process. Depending on the experiment this solution might require a large user base. Moreover, this solution might not scalable with a large number of experiments in parallel.

Measure and keep track of the experiment conditions.

Description: in this solution, the experimentation manager feature decides if the systems can experiment or not.

Design rules: the experimentation manager needs to keep track of the context and determine the current state of the system.

Design constraints: the system is not allowed to experiment without determining complying with the experiment boundaries.

Consequences: this requires the system a capability of keeping track of its context and business goals. Incomplete or wrong information of the context leads to experiments running outside its boundaries.

Pros: each experiment is independent. The experimentation manager can stop the experiment if the system is running out of the experiment boundaries.

Cons: the system needs to instrument several context variables to correctly induce the current state. This can create a large overhead for the first experiments in an organization that is not data-driven.

Decision: The decision is made to measure and keep track of the experiment conditions. In dynamic systems and in uncertain environments the system there are no guarantees that the system will stay in its initial experiment boundaries. Traditional controlled experiments systems rely on the R&D teams to check the conditions of the experiment. Manually keeping track of the experiments limits the number of experiments the R&D team can run in parallel.

4.6 Architecture framework

In this section, we describe an initial architecture framework that satisfies the identified qualities to support automated experimentation. This framework was developed based on the architecture analysis and in the of the architectural design decisions provided in the previous sections.

4.6.1 The architecture framework

The presented architecture framework is represented in Figure 4.2. This architecture is the result of the design decisions over several iteration processes and inspired by the analyzed architectures described in previous sections.

The intention of this architecture framework is to provide an existing system the capability of doing automated experiments decentralized in feature level, rather than providing an architecture for the whole system. Manual hypothesis testing is integrated as the hypothesis are still formulated by the R&D team. The presented architecture modules are described next

4.6.1.1 Monitor

This module is responsible for the data collection. The data collected come from the probes available in the system and in the SuE (system under experimentation), therefore both local and global behavior. This module is directly related to the data collection in the discussed qualities. Access to all the necessary data for experimenting requires proper instrumentation of the code. This module does not represent only a stream of raw data into the automated experimentation architecture framework. It rep- resents data processed that add information to the system. Effector. This module is responsible for interfacing with the managed system. Besides the monitor, it is the only point of contact with the rest of the SuE. This module requires that the managed system expose interfaces for interaction with the system. This concept of not intermixing the experimentation code and the managed system code is directly related to the external adaptation quality. The same observations made to the monitor module are valid for the effector.

4.6.1.2 Experiment coordinator

This module is responsible for running the experiment and coordinating with the version manager. This module controls only the specific SuE, other experiments have their own experiment coordinator modules. The experiment coordinator can control experiments such as A/A, A/B/n, explore/exploitation and crossover experiments. This module keeps track on when to experiment, the number of experiments that should be run, which solution is more significant. This module receives inputs from the conflict-list manager if it is allowed to run an experiment or not. It also receives inputs from the experiment watchdog module, if the system is deteriorating any global metrics if it went out of boundaries or if it still needs to perform more experiments.

4.6.1.3 Version Manager

This module is responsible for man- aging and generating different versions to experiment. This can be acting in parameters or replacing whole sub- component models. The version generator keeps a list of the versions used and accepts versions inputs from the Knowledge Exchange module and the Version Generator. This allows the system to experiment both automatically generated versions, as well as manual versions crafted by the R&D team. Although this module is not directly connected to a one of the design decisions listed, this module is linked to both the functional and quality requirements

4.6.1.4 Version Generator

This module can accommodate different artificial intelligence algorithms that we might want to test. The generation algorithm is not specified, but it could include machine learning algorithms, such as reinforcement learning algorithms, genetic algorithms, parameter scheduling or randomized versions. This module is directly connected to the learning quality.

4.6.1.5 Experiment Watchdog

This module checks the conditions that the system can run the experiments, such as when the system should continue experimenting and when it should stop. If the system goes out the

predefined boundaries or if there is deterioration in global metrics this module can stop the experiment and return the system to the "safe" version. Having a stop condition for global metrics prevents the system improving a local metric but degrading a global metric. If any of the stop conditions is reached this module signals to the experiment coordinator to stop the experimentation process or to roll back to a safe version.

4.6.1.6 Conflict-list manager

This module keeps track in run- time of components that are being experimented with and which factors it affects. This is an important component in a decentralized experiment environment. Several other systems of the robot can be experimenting. This manager keeps track of those systems in order to avoid confounding variables in the experiment. This is directly related to the decentralized adaptation quality. In a generic implementation, the conflict manager advertises its current state (experimenting or not) and listen to other conflict manager's states.

4.6.1.7 Metric Analysis

This module is responsible for keeping track of the managed system behavior and the value function. This module serves as a trigger to drive the adaptation through optimization or through keeping track of the validation process. In this module, we insert the value function and we run our statistical analysis. This module is directly related to the performance reflection quality.

4.6.1.8 Knowledge and Information Exchange

This module communicates with the optimization and experiment validation module and with the external world. This module is responsible for sharing discovered solutions in the experimentation process and also for sharing and learning the validated solutions from the experiment through a central server infrastructure. This also represents a way in which the R&D can interact with the system, either helping in the analysis step or proposing different versions

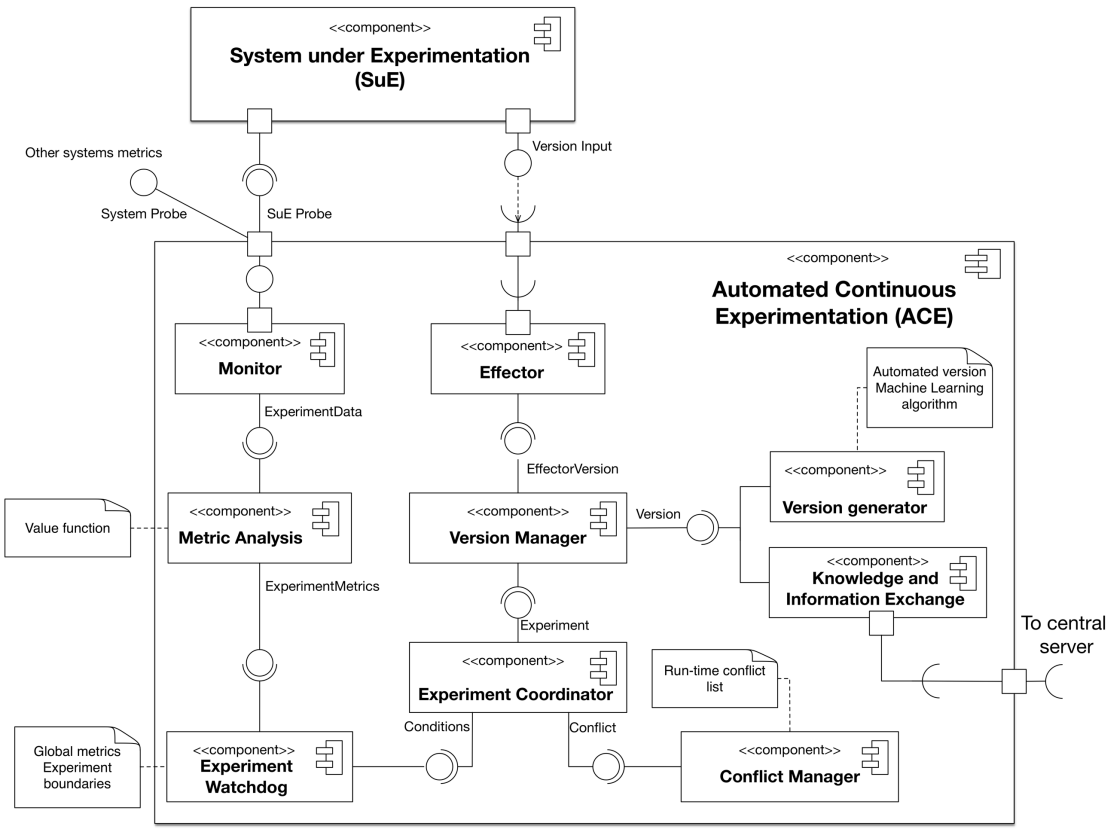

Figure 4.2 - The architecture framework for automated continuous experimentation

not generating by the version manager, for example, testing different algorithms. This module is directly related to the Knowledge Exchange quality.

4.7 Automated experiments in a human-robot interaction problem

In this section, we present an experimental implementation and evaluation of the automated continuous experimentation architecture framework. In this experimental scenario, we first validate the correctness of the architecture behavior. Second, our architecture for automated experimentation indicates a more cost-effective solution for running experiments compared to manual experimentation.

4.7.1 Proxemics distance in Human-Robot Interaction

Human interaction is based on several unwritten and subjective rules. One example is respecting other people's personal space. In human-human relations, several social factors play an important role in this interaction. Not conforming to these rules may cause miscommunication and discomfort. To have a good human-robot interaction, the robots must follow similar rules [100]. The large body of work in Human-Robot Interaction (HRI) identified several factors that come into play in proxemics distance, such as gender, age, personal preferences, technology involvement, crowdedness, the direction of approach, form factor and size [100]. Different works recognize some base distances and how they are influenced depending on a change of factor. However, this is still an open problem. The development of new robots and the deployment of these robots in very different contexts (e.g. different countries) require new manual experiments to validate and optimize the proxemics distance. The presented automated continuous experimentation framework can be used to allow the robot to try different distances and learn an optimal distance on the context that it is inserted. This would allow robots to adapt its proxemics distance and validate it without the need of designing costly experiments for each different context.

The framework was instantiated in a research mobile robot Turtlebot[1] using the Robot Operating System (ROS) middleware[2]. This platform is similar in size and form factor with several commercial mobile companion robots.

The source code for the full automated experimentation framework implementation is available at https://github.com/davidissamattos/david_ws. Each of the component blocks represented in Figure 4.3 were implemented as a separate Python process communicating through publish-subscribe messages. In this instantiation, the conflict manager and information exchange components were not implemented as the system is only running one experiment using only one platform robot.

The instantiated architecture consists of six ROS nodes that can be mapped to the architecture framework modules. Each node was implemented as a separate process, communicating through a mix of publish-subscribe and client-server methods as defined by ROS. Figure 4.3 shows the instantiated framework. In this initial step of the research we are focusing on implementing the framework in one system initially, therefore the knowledge exchange quality was the only one not contemplated in this experimental validation.

4.7.1.1 Feedback monitor

This node can be mapped directly to the monitor module. It is responsible for capturing the human feedback. In this case, we receive input from both verbal feedback (e.g. "Too far") and non-verbal feedback (e.g. if it is too close the person steps back).

[1] www.ros.org
[2] www.turtlebot.com

4.7.1.2 Metric Analysis

This node receives as input the value function of the system and analyzes input data. The value function we are using is the user satisfaction. We expect our user to be satisfied at least 70% of the approaches in the long run (after a minimum number of experiments).

4.7.1.3 Experiment Watchdog

This node verifies the current state of the system and the boundaries of our problem. For this system, we defined some boundaries such as, do not get closer than 20 cm from the human, do not experiment if the battery is low and if the experiment is performing poorly (e.g. less than 30% of the cases) we rollback to another version. This module works even if the version manager generates values outside the constraints of the experiment (by implementation/runtime errors).

4.7.1.4 Experiment coordinator

This node is responsible for keeping track of which experiment is going on, optimization, validation or running in static loop mode.

4.7.1.5 Version manager

The version manager node receives input from the experiment coordinator regarding which experiment is running. It the feature is running in safe- mode it uses a safe predefined distance. If the learning process has converged the experiment coordinator requests a static learned version. The version manager generates these versions either by static input or by calling a learning module to generate it (e.g. calling the machine learning module).

4.7.1.6 Version Generator

the version manager requests new versions to the machine learning model in the version generator. This module uses the K-means clustering algorithm with the k-means++ initialization algorithm implemented in the scikit-learn[3] Python library. This learning algorithm is a particular solution of the k-armed bandit optimization problem.

4.7.1.7 Effector

this node is responsible for interacting with the human approach feature and modify in runtime its internal value for approaching. In the ROS context, this means changing parameters in the parameter server. The implemented architecture framework is completely external to the system. The system runs normally without the framework and even with if the instantiated framework crashes.

4.7.2 Experimental results

In this subsection, we describe the experimental conditions which we tested this system. The robot was placed in an office environment. One participant took place in the experiment. The participant was instructed to give verbal, non-verbal feedback to the robot if they think the robot is too far or too close (feedback value -1), and not to give any feedback if they think the robot is at a comfortable distance (feedback value 1). The robot always moves to a different location (away from the participant) before approaching again. The participant did not have any previous experience with autonomous robots or knowledge of the internal system of the robot. The participant also didn't know that the robot was learning from the given responses. Situations, as when the system goes out of the defined boundaries, were tested manually during the experiment (by explicitly sending the system to a new state) without the knowledge of the participant. All cases were handled by the stop experiment module without any problems.

[3] http://scikit-learn.org/

Figure 4.4 shows three graphs representing the learning process of the system experimenting with different uniform random approaching distances. The first graphic shows the system exploiting the solution space trying different distances. The second graphic shows the system learning and refining the lower and upper distance boundaries using the clustering algorithm. The third graphic shows the system learning a static distance after several robot approaches. The static learned distance is the centroid of the cluster located between the boundaries. This graphic shows an online optimization through experiments on the proxemics distance. Changes in the user behavior are reflected in the learning process. Although this experiment shows only one experiment in an office environment, this could be extended to incorporate different factors that influences the robot behavior, such as incorporating the type of room in the experiment or identifying different use scenarios. This would allow the robot to continuously experiment and learn new distances in different contexts, therefore increasing the value it delivers. This experiment suggests a more effective way of experimenting compared to manual experimentation. Each experiment requires time to run and set up and are valid in restrictive conditions. Although this experimenting method does not guarantee optimality, manually experimenting a matrix of solutions can be expensive in both terms of set-up cost and time.

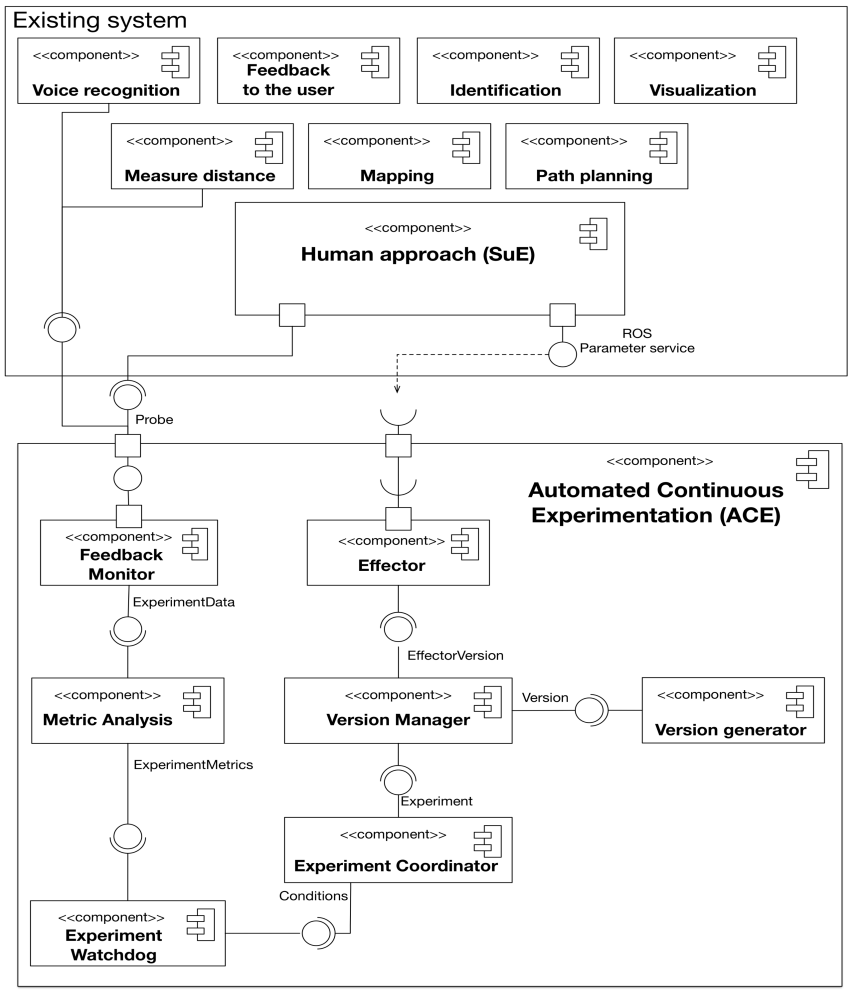

Figure 4.3 - Instantiation of the automated continuous experimentation framework in the Human-Robot Proxemics distance problem.

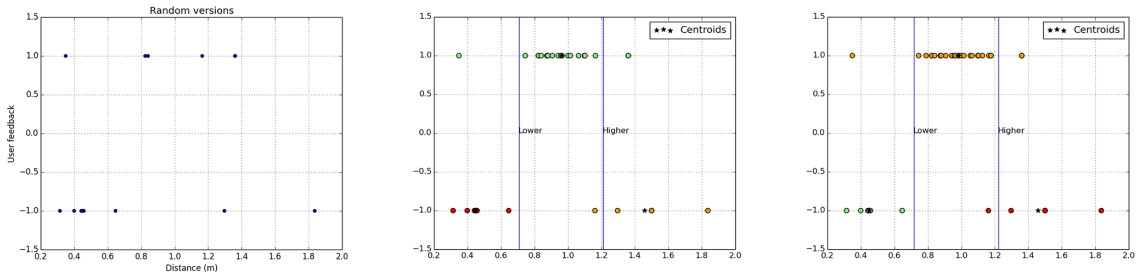

Figure 4.4 - Three graphics representing the learning of the system. The first graphic shows the robot exploring the feedback-distance space. The second shows the robot learning an interval and refining it. The third shows the system after achieving a validated static value for the distance. This distance is represented by the centroid in the orange section.

4.7.3 Cost-effectiveness

Kohavi [6] provides guidelines on a minimum number of samples when running A/B experiments. Using these guidelines, the optimization of the proxemics distance for one individual requires at least 96 samples for each variation (assuming a baseline of 50% with a minimum detectable effect of 20%, with a statistical power of ß=80% and significance level of α=95%) [6]. Considering a grid of 1 meter per sample in a range of 1 m to 5 m, it results in 480 samples. Using the automated continuous experimentation framework, the system converged to a distance in 37 samples. Integrating this result with a traditional A/B experiment results in a total of 133 samples to achieve statistical validity in the optimization process. Although this result was not generalized, it already suggests a more effective way to run experiments.

4.8 Conclusion

Experiments in the field are used in a problem-solving process to drive both innovation and optimization of post-deployed systems. Companies are moving towards to experiment-based development, where experiments support the decision-making process. Several challenges, such as resources, the experiment architecture and novel engineering approaches arise when running experiments in a large scale. In this chapter, we study automated experimentation to address these challenges.

This chapter takes the software architecture perspective to understand and develop an initial architecture framework that supports both hypothesis validation and for optimization through experiments. Different architecture qualities are identified and it is proposed an initial architecture framework to support automated experiments.

The architecture framework is evaluated using an autonomous mobile robot establishing the optimal human-robot proxemics distance. The robot implements an optimization solution based on the explore/exploitation problem running automated experiments. The experimental validation not only assesses the correctness of the framework behavior but also suggests that this is a cost-effective way to run experiments.

4.8.1 Research Challenges

Automated experiment systems still have a long way to go before they are matured. Different domains develop solutions and algorithms that continuously push systems in the conceptual solution proposed in [27]. However, these different domains solve their experiment-specific problem without a unified view over the experimentation process itself. This work brings together different facets of automated experimentation and proposes a framework to support automated continuous experimentation. The framework was validated experimentally in the context of an autonomous system and is currently under validation in the mobile domain and in web-systems. One research challenge is to evaluate this and other architecture frameworks in

different contexts. This will bring into perspective the challenges from the different domains that might reflect in both the desired architecture qualities and in the architecture framework.

A second research challenge is to combine manual experimentation with automated experimentation in R&D teams as part of the development process for the product. As much as the culture for experimentation changes the organization perspective [76], automated experimentation can change how the systems are developed.

A third research challenge in the process of fully automate experiments is the ability to formulate hypothesis automatically from the data. Manual experimentation re- quires significant effort for the hypothesis formulation from analyzed data. The proposed framework automates how to run and evaluate experiments, but it still does not support hypothesis generation. Creating hypothesis automatically from recorded data can leverage the amount of experiments the system can run and generate deeper insights on how the system works in the uncertain environment. This would allow the experiment innovation perspective to be an integral part of the system.

Concluding, although there are several challenges ahead, we are moving to a future where systems get better every day we use them through automated experimentation.

5 Experimentation in Embedded Systems

This chapter is based on the following publication:

Challenges and Strategies for Undertaking Continuous Experimentation to Embedded Systems: Industry and Research Perspectives
David Issa Mattos, Jan Bosch and Helena Holmström Olsson
In: Garbajosa J., Wang X., Aguiar A. (eds) Agile Processes in Software Engineering and Extreme Programming. XP 2018. Lecture Notes in Business Information Processing, vol 314. Springer, Cham

Chapter Summary
Continuous experimentation is frequently used in web-facing companies and it is starting to gain the attention of embedded systems companies. However, embedded systems companies have different challenges and requirements to run experiments in their systems. This chapter explores the challenges during the adoption of continuous experimentation in embedded systems from both industry practice and academic research.

This research was conducted in two parts. The first part is a literature review with the aim to analyze the challenges in adopting continuous experimentation from the research perspective. The second part is a multiple case study based on interviews and workshop sessions with five companies to understand the challenges from the industry perspective and how they are working to overcome them. We found a set of twelve challenges divided into three areas; technical, business, and organizational challenges and strategies grouped into three categories, architecture, data handling and development processes.

The contribution of this chapter is twofold. First, it identifies the key challenges faced by embedded systems companies when adopting continuous experimentation. These challenges are identified from both the industry perspective, through a multi-company case study, and the academic perspective, through a literature review. Second, we propose different strategies and guidelines to overcome the identified challenges. To the best knowledge of the authors, this is the first research to present an extensive set of challenges and strategies that embedded systems companies face when adopting continuous experimentation. Moreover, the analysis of the challenges points out the need for new tools and novel solutions for the further development of experimentation in embedded systems.

5.1 Introduction

Traditional embedded systems companies continuously rely on software to be a differentiator on their products. As the software size of the products increases, these companies are moving from being mechanical producers to software companies. In their development process, these companies traditionally make use of up-front requirements and rigid methodologies to ensure quality or safety attributes in their products. Nevertheless, the requirements of several parts of their systems are not clear or cannot be defined in advance [101]. In this context, developers either negotiate with requirement teams or they make implicit assumptions about the requirements [102].

Even during the requirement specification, several requirements are written based on assumptions and does not necessarily deliver value to the company or the customers. Often, research and development effort is spent on features that are never or rarely used [63] by the

users of the product. To minimize the full development of features that do not deliver value, companies make use of post-deployment data of current products to iterate in future software releases or in even in new products. In the web domain, companies provide empirical evidence of the use of continuous experimentation in their development, decision-making and feature prioritization process [6], [23], [26].

As software becomes the key differentiator for many embedded systems companies, these companies started to adopt continuous development practices, such as continuous integration, deployment, and experimentation to develop faster, better and more cost-effective products. A typical pattern that companies follow is shown in the "Stairway to Heaven" model [103]. When these companies start to move to move to continuous deployment scenarios, they see opportunities to run their first experiments as well.

Although the research in continuous experimentation in web systems is continually growing, there are few examples of works investigating the use of continuous experimentation in embedded systems.

Giaimo and Berger [104], discuss continuous experimentation in the context of self-driving vehicles. The paper presents functional (such as instrumentation, logging, data feedback to a remote server) and non-functional (separation of concerns, safety, short cycle to deployment) requirements to achieve continuous software evolution. Bosch and Olsson [27], extended the concept of experimentation towards automated experimentation. Automated experimentation aims to leverage the number of experiments by letting the system own and control the experiments, opposed to the R&D organization. Mattos et al. [105], [106], identified a set of architectural qualities to support automated experimentation that was implemented in a research mobile autonomous vehicle.

To the knowledge of the authors, the first research discussing the experiments in embedded systems appeared in 2012 [107]. This chapter discusses experimentation in the context of Innovation Experiment Systems. It identifies some challenges with experimentation in embedded systems, such as experimentation in safety systems, managing multiple stakeholders and hardware limitations. It also presents an initial infrastructure to run experiments in embedded systems

This chapter identifies and analyzes the different challenges that embedded systems companies face when adopting continuous experimentation in their development processes. Moreover, it also presents strategies, guidelines, and potential solutions to overcome each of the identified challenges.

The scope of this research is captured with the following research question.

RQ: How can embedded systems industry adopt continuous experimentation in their development process?

This research question is further developed in terms of the following sub-questions:

RQ1: What are the recognized challenges towards continuous experimentation faced by the embedded systems industry?

RQ2: What are the recommended strategies to facilitate the use of continuous experimentation in the embedded systems domain?

5.2 Research method

The research process used in this study combines a literature review with multiple case study. This research method aims to strengthen the evidence of the challenges and strategies found in a multiple case-study with others found in the research literature. Research in continuous

experimentation generally utilizes the case study as the research method, combining results from both approaches reinforce the empirical evidence of the findings.

The method is composed of two parts. The first part consists of a literature review in the continuous experimentation domain. This literature review collects challenges and strategies to overcome them from academic research. The second part consists of semi-structured interviews with software companies in the embedded systems domain. It aims to be exploratory, collect and confirm challenges and strategies from the embedded systems industry. Below, the research method is described in details. The results of both parts were aggregated and described in Section 5.3. Table 5.1 summarizes the research process.

5.2.1 Literature Review

The first part of the research method consists of a literature review in continuous experimentation. Although most of the studies in continuous experimentation focus on web-facing companies, the experiences from this domain, sometimes, can be extrapolated to the embedded systems domain. In this literature review, the authors identified challenges recognized in academic collaboration with industry, regardless of the industry domain. The identified challenges were discussed with the embedded systems companies to see if the literature challenges were also relevant in this domain.

Table 5.1 - Summary of the research method. LR stands for the literature review part and CS for the multiple case study part.

Step	Description
1	Search definition and execution (LR)
2	Papers review (LR)
3	Identification of literature challenges and strategies (LR)
4	Data selection: Contact with companies (CS)
5	Semi-structured interview protocol definition (CS)
6	Data collection: Interviews and workshop (CS)
7	Data analysis: thematic coding and categorization (CS)
8	Case study report (CS)

Relevant works in the literature covering continuous experimentation were identified by searching the Scopus digital indexing library, by keywords, tile and abstract. The used search phrase was:

```
((continuous experimentation) OR (field experiments) OR (innovation experiment systems))
AND (software engineering)
```

This search query was restricted to the fields of engineering and computer science and limited from 2000 to 2017. This search phrase resulted in 534 articles. Positioning papers and papers with less than 5 pages were excluded. From this subset of articles, the results were filtered based on the abstract. After the first screening process, the papers were read in their integrity. Continuous experimentation is also largely studied from the statistical/algorithmic side. Research papers that focused solely on improving or evaluating algorithms without industry evaluation or application were excluded.

After this screening process, the authors identified 30 articles with relevance to this study. An additional set of 12 articles were included using a snowballing [20] process, where new references were added according to the references mentioned in the other articles. Thematic coding was used to [71] identify the challenges from the literature. These challenges were categorized according to the three different categories of the Experimentation Evolution Model

[26] discussed in Background (Chapter 2), the technical, the organizational and the business perspective. The identified set of challenges were also used as input for the semi-structured interviews. The strategies are categorized in three groups: changes in the development process, changes in the system's architecture and changes in how the experiment and organizational data is handled and analyzed.

The complete set of papers can be found at the following link https://github.com/davidissamattos/public_documents/blob/master/LR-XP18.png.

This part of the research process allowed the identification of challenges that served as input for the multiple case study and confirmation of identified challenges inside the company.

5.2.2 Multiple case study

The second part of the research method consists of a multiple case study [71] with semi-structured interviews conducted with software companies in the embedded systems domain. This study was conducted from December 2016 to October 2017 with five companies in the embedded systems domain.

The empirical data consists of interviews and a workshops transcripts and notes. There were 8 individual semi-structured interviews with an average of one hour each, three in *Company A,* two in *Company B*, one in *Company C,* one in *Company D* and 2 in *Company E*. The workshop session was conducted with 8 people from *Company A* lasting 3 hours. The analysis of the empirical data consisted of thematic coding of [71] interviews transcriptions and notes to identify and categorize the challenges and solutions. Additionally, during the interviews challenges identified in the literature were clarified to the interviews and asked if the current company relates to the challenge partially or not.

The empirical data were aggregated together with the identified challenges and strategies from the literature review. The current published research already provides guidelines and solutions for the challenges that were also identified in the literature review phase. Other guidelines and solutions were suggested by practitioners during the interviews. Challenges identified in the literature that was not confirmed neither through a previous case study nor by the case study companies are not shown

Due to confidentiality reasons, only a short description of each company and their domain is provided:

Company A is a multinational conglomerate company that manufactures embedded systems and electronics and provides software solutions for both consumers and professionals. This study was conducted with two teams, one providing mobile communications solutions and the other providing business-to-business products. In recent years, the company started to adopt experimentation in their software solutions and is looking for data-driven strategies in their embedded systems products. The interviewees were developers, managers and data analysts.

Company B is a multinational company that provides telecommunication and networking systems. The company is adopting continuous development practices and is looking for new strategies to deliver more value to their customers by optimizing their products. The interviewees were managers.

Company C is a global automotive manufacturer and supplier of transport solutions. As the company's products are continuously growing in complexity and software size, the company is looking for strategies to prioritize their R&D effort and deliver more value to their customers. As some employees have experience in web and pure software-systems development, experimentation is getting attention in some development teams. Challenges in experimentation

arise since the company is subjected to several regulations and certification procedures. The interviewee was a senior engineer.

Company D is a global software company that develops and provides embedded systems software solutions related to autonomous driving technology for the automotive industry. Autonomous driving is an emerging and fast-moving technology and the company is looking to deliver competitive solutions faster by adopting continuous development practices. However, as it interfaces with the highly regulated automotive domain its software is also subjected to regulation and certification. The interviewee was a manager.

Company E is a global software company that develops both software and hardware solutions for home consumers. The company already has experience running continuous experimentation in their web systems and is starting to run experiments in their hardware solutions. The interviewees were senior data analysts working in experimentation in their embedded systems.

5.3 Challenges and proposed strategies

This section presents results obtained from the research process. The challenges are grouped in the three different perspectives as discussed in the Experimentation Evolution Model [26]: the technical challenges, the business challenges and the organizational challenges. The technical challenges refer to challenges related to the system architecture, experimentation tooling and development processes. The business challenges refer to challenges faced in the business side, such as evaluation metrics, business models and privacy concerns. The organizational challenges refer to challenges faced by the cultural aspect of the R&D organization.

All the strategies identified in this study are used, suggested by companies, or supported by strategies identified in previous literature case studies. The strategies are categorized in three groups: (1) changes in the development process. This refers to how companies organize their development activities. (2) changes in the system's architecture. Often restrictions in the running experiments comes from limitations in the system's architecture, that does not support data collection, or does not allow parametrization of features for experiments. (3) changes in how the experiment and organizational data is handled and analyzed. This refers to how the company stores data, comply to data regulations or use data analysis tools. The challenges are not presented in any specific order as they might reflect different challenges the companies are facing.

Figure 5.1 represents a summary of the identified challenges and strategies. In Figure 5.1 it is possible to see the relation of how each strategy relates to the different challenges, as some of them are part of the strategy of one or more challenge. This figure was obtained using the thematic codes generated in the analysis of the interviews. It maps the identified challenges within their groups with the obtained strategies groups. The rest of this section discusses each challenge individually and presents strategies to overcome them.

5.3.1 Technical Challenges

5.3.1.1 Lack of over the air (OTA) updates and data collection

Continuous experimentation requires over-the-air (OTA) post-deployment data collection and updates. When testing a different hypothesis, the system needs to have the ability to measure the specific behavior under investigation and to update the system with the new variants as well. It is possible to run experiments without OTA, however, several experiments pitfalls can be

Figure 5.1 - Summary of the challenges and the strategies faced by embedded systems companies adopting continuous experimentation.

identified in the first hours and be corrected [6]. Moreover, experiments for optimization are looking in practical significance as low as 1-2% in their metrics [6], [108]. If OTA updates and data collection are not available the cost of the experiment and the practical significance level are high and the optimization process might not be worth it.

Strategies: At the moment of this study, embedded system companies are not looking into experimentation in low level systems, but in computing systems that already support modern operating systems with connectivity and the necessary infrastructure for OTA updates. OTA updates and post-deployment data collection should be part of the functional requirements of the system when designing the hardware. Mobile companies already provide such functionality in their operating systems. Car manufacturers are also introducing continuous delivery of new software to their vehicles in the context of autonomous vehicles (Tesla Motor's Model S, Volvo Drive Me and the Volvo XC90).

5.3.1.2 Lack of experimentation tools that integrate with their existing tooling

Continuous experimentation started in web-facing companies. Today several experimentation tools, both commercial and open source, are available on the website and mobile applications domains. However, in the embedded systems domain, companies lack tools that integrate with their development process. Setting up an infrastructure to run experiments from scratch increases the cost of running the first experiments while hindering the benefits.

Strategies: Several tools available for websites are open source or have open source SDKs. Although not ideal, some of these tools can be modified to support experimentation problems. Experimentation-as-a-Service (EaaS) is a business model that provides a working platform for continuous experimentation. EaaS have the benefit of avoiding the cost and pitfalls of development of an experimentation platform from scratch. EaaS platforms also provide SDKs that can be incorporated in the product [109]. However, the system under experimentation should support data collection so it can be integrated with EaaS tools.

5.3.1.3 Expensive testing environments

Software-intensive embedded systems are extensively tested before release. One of the challenges faced by embedded systems companies is to include experimentation as part of the verification and validation process. In some cases, such as in the development of a new vehicle,

the testing environment is expensive and not all experiment hypotheses are allowed to go to a physical testing platform. This high cost also increases minimum level necessary to reach practical significance and demotivates teams to formulate hypothesis beyond the basic requirements of the system.

Strategies: the development of experiments in the embedded systems domain require additional steps from the hypothesis to the final user. The development of a feature in embedded systems follows a testing procedure, beginning with integration and going to simulation, test beds, internal deployment until user deployment. The experimentation procedure should follow similar testing procedure, to identify early pitfalls, and even improve the system behavior during each testing phase.

The practical significance level to implement a new hypothesis increases with the associated costs of such testing procedure. The EDAX model [27] describes how experimentation and automated experimentation is incorporated in this process. Automated experimentation [105] also suggests that it can reduce the experimentation costs and therefore the practical significance level.

5.3.1.4 Experimentation constraints in real-time and safety-critical systems

Embedded systems are employed in several real-time and safety-critical systems. These products have subsystems that are constrained to regulations and certification. Experimenting with these systems in the field might not be allowed by regulation or might impact substantially the performance of the system.

Strategies: Embedded systems companies are starting to run their first experiments. Safety-critical or real-time systems provide additional challenges, as it is subjected to legislation and certification. The initial recommendation in all case study companies is not to run experimentations in these subsystems. However, these safety-critical subsystems can run experiments in the earlier phases prior to the deployment, as discussed in the EDAX model [27].

5.3.2 Business Challenges

5.3.2.1 Determining good experimentation metrics and metrics validation.

One of the biggest challenge faced by companies is to determine good business metrics to understand and compare different experiments, and validate that the current metrics are aligned with the company strategy

Strategies: Web companies traditionally rely on conversion metrics such as Click-Through-Rate in the beginning of their experimentation process. As their experimentation teams and the number of experiments increase the metrics start to become more tailored to the business and stable [26]. Embedded systems companies can have very different and complex metrics, depending on the product. However, team level optimization experiments can use customized metrics. Olsson and Bosch [110] presents a systematic approach to determine metrics and value functions for experiments. This is an iterative process that should be refined with usage and aligned with the business strategies and goals. As the metrics become complex, companies allocate of resources for the evolution and ensuring that the experiment metrics are aligned with the company's main KPIs.

5.3.2.2 Privacy concerns regarding user data.

Continuous experimentation relies on the collection and analysis of post-deployed software. However, some issues arise when collecting data, such as the legal and contractual issues or user consent and data sharing.

Strategies: Data sensitivity and the use of data vary largely between different organizations and countries. Data collection should be aligned with the legal requirements for utilization and consent of the users. Data regulations such as the European GDPR (https://www.eugdpr.org/) create restrictions that might imply in technology and process modifications for compliance. Additionally, some ethical questions regarding the experiment must be evaluated, such as: How are participants guaranteed that their data, which was collected for use in the study, will not be used for some other purpose? What data may be published more broadly, and does that introduce any additional risk to the participants? Web companies, besides compliance with regulations also create ethical checklists to ensure that the experiments follow the companies' policies [111].

5.3.2.3 Lack of sharing user data in business-to-business (B2B) solutions

Several embedded systems companies operate in a business-to-business domain. In this scenario, there is a difference between user and customer data. Experiments with users might not be possible, they might require deeper involvement between the companies, or there might be a mismatch between the customer and the user value [101].

Strategies: Ecosystems refers to companies co-opting third parties to build and leverage their products and services in such a way that they have more utility value to their customers [112]. In this sense, companies might agree on implementing and sharing data collected inside the ecosystem. Some mobile operating systems (e.g. iOS and Android) collect data and usage statistics to share with app developers. Although most of its use is connected to crash reports, similar strategies can be used to share user data in business-to-business products.

5.3.2.4 Lack of insights obtained from the collected data

Companies are continuously collecting data from their deployed software. The collected data is mainly used for troubleshooting purposes. However, little insight is provided by the collected data [110]. In the Experimentation Evolution Model [26], web companies evolve from centralized data science teams to small data science teams presented in each product teams. The interviewed embedded systems companies don't have data science teams incorporated in the product development.

Strategies: If the experimentation benefits are not clear, the extra cost of involving data scientists in the product development might be a large step. Different companies started to provided experimentation and data analysis services. Experimentation tools usually incorporate basic statistical analysis, such as statistical significance testing, power analysis, A/A tests and more. Using experimentation and data analysis services to generate basic insights can be used as a short-term solution. Once the benefits of experimentation are clear to the company, investments such as integrating data scientists in the product development or acquiring a complex tool are easier to justify.

5.3.2.5 Long release cycles

Traditionally, embedded systems have a long software release cycle based on upfront defined requirements. Sometimes the software is deployed only once and last for several years [101], [107]. This happens due to several reasons, from both the organizational (structure and decision-making) and business (engineering effort in every cycle, requirements definition and products updates) to the technical perspective (architecture, functionalities available and support for over-the-air updates).

Strategies: From the organizational and business perspective, continuous experimentation aligns with the organizational transition to agile methodologies and the Lean Startup methodology [42]. Continuous experimentation makes use of extreme programming practices such as continuous integration, delivery and deployment to deliver experiments and new

software aligned with customer behavior. The Stairway to Heaven [103] conceptual model helps companies to evolve their practices towards continuous deployment of software.

5.3.3 Organizational Challenges

5.3.3.1 Managing multiple stakeholders in the experiment design

One of the challenges embedded systems companies face is the involvement of multiple stakeholders in an experimental design. Experimentation in embedded systems requires that the involved stakeholders understand the implications of continuous practices in their systems.

Strategies: Embedded systems require the interaction with multiple stakeholders, such as software developers, systems architects, electrical and mechanical engineers, suppliers and subcontractors. Continuous experimentation requires that these stakeholders are aware of the implications in the system design. To overcome some of these challenges, it is prosed a decoupling of the application and the underlying software and also a decoupling in time (software is not integrated at the manufacturing time) [107]. Additionally, if the interaction of the stakeholders happens in a business ecosystems perspective the experiment can be designed to benefit multiple parts [112].

5.3.3.2 Highest Paid Person Opinion - HiPPO

Some companies are organized in vertical structures, where lower rank developers have fewer possibilities to influence and address customer's needs. Several requirements and architecture specifications are based and determined by higher paid ranks inside the company.

Strategies: This challenge is persistent in several domains and it is not restricted to the embedded systems industries. This challenge is discussed extensively in [6] among other publications. The traditional adopted strategy is to run the first experiments. Usually, experiments continuously disprove beliefs and opinions adopted by the higher paid ranks [6]. However, this requires changes in the organizational and cultural aspect of the company.

5.3.3.3 Tuning experiments is repetitive and requires highly qualified engineers

One of the interviewed companies runs experiments for parameter optimization. The experiments rely on the system response instead of the customer response. However, running these experiments for tuning and optimization is a repetitive task that consumes R&D time and requires highly qualified engineers to perform them.

Strategies: Existing algorithms in search-based optimization, reinforcement learning and others artificial intelligence algorithms support this kind of optimization strategies. However, both the complexity of these algorithms as well as the introduced technical debt in the existing systems [77] prevent embedded systems companies to use such strategies. Experimentation-as-a-Service solutions allow companies to test Machine Learning algorithms in their system for optimization purposes. Although still in early phases, automated experimentation [105] solutions can help companies to optimize their systems through field experiments.

5.4 Validity Threats

The first threat to the validity of this study refers to the scope of the literature review. The search query was applied to the Scopus indexing library. Both the choice of the search string and the indexing library could miss other research work that can contribute to the literature review. To mitigate this threat the authors performed a backward and forward snowballing [20] process. The snowballing process allowed the authors to identify other cited work in the same area that was not identified by the search query.

An external validity to this is study is the generalization of the challenges to the entire population of embedded systems companies. To mitigate this threat, the authors sample companies producing different products in embedded systems. The authors sampled contacted

multiple companies explaining the research goal and selected only companies that are adopting/running controlled experiments in their development process were included. During the data analysis part, we reviewed all challenges only challenges that had correspondence in more than one company or that could be triangulated with the literature review were included. Challenges that could not be triangulated with other source, and that could be specific to current situation of the company, were not included in this study.

The companies that participated in this study are adopting their first steps towards continuous experimentation and are running their first experiments or trying to scale experimentation practices from a few development teams to the organization. Therefore, most of the presented challenges are faced in these first steps and cannot be generalized to companies or teams that are running experimentation at scale. As the companies evolve their experimentation practices, new challenges will arise from all three perspectives.

5.5 Conclusion

This chapter addresses the question of how embedded systems companies can adopt continuous experimentation in their software development process. This question can be divided in two parts: first, the identification of problems and challenges that limit the adoption of continuous experimentation, and second selected strategies adopted by companies to overcome these challenges.

We identified twelve key challenges faced by embedded systems and then grouped in three perspectives, the business, the technical and the organizational. The challenges are also presented with suggested strategies to overcome them. The set of strategies can be grouped in three categories, changes that need to take place in how the company handles and analyze the post-deployment collected data, changes in the company development process and changes in the product architecture. The relation between the different strategies and the challenges is seen in Figure 5.1. The paper used a combination of literature review and a multiple company case study to provide a stronger empirical evidence.

Further research is needed to understanding how the system can be architected to support continuous experimentation as a first-class citizen in the development process while still guaranteeing safety and real-time requirements as well as intermittent connectivity. Additionally, continuous experimentation changes how the development process takes place, as it emphasizes in an outcome-driven development and this scenario might lead to impactful organizational changes. For future works, we are investigating where is the perceived highest return on investment that companies see and plan to invest to overcome the identified challenges and further support of continuous experimentation in their products.

6 Optimization experiments in the continuous space

This chapter is based on the following publication:

Optimization Experiments in the Continuous Space: The Limited Growth Optimistic Optimization Algorithm
David Issa Mattos, Erling Mårtensson, Jan Bosch and Helena Holmström Olsson
In the Proceedings of the 10th International Symposium on Search-Based Software Engineering, Montpellier, France, 2018

Chapter Summary
Online controlled experiments are extensively used by web-facing companies to validate and optimize their systems, providing a competitive advantage in their business. As the number of experiments scale, companies aim to invest their experimentation resources in larger feature changes and leave the automated techniques to optimize smaller features. Optimization experiments in the continuous space are encompassed in the many-armed bandits class of problems. Although previous research provides algorithms for solving this class of problems, these algorithms were not implemented in real-world online experimentation problems and do not consider the application constraints, such as time to compute a solution, selection of a best arm and the estimation of the mean-reward function.

This chapter discusses the online experiments in context of the many-armed bandits class of problems and provides three main contributions: (1) an algorithm modification to include online experiments constraints, (2) implementation of this algorithm in an industrial setting in collaboration with Sony Mobile, and (3) statistical evidence that supports the modification of the algorithm for online experiments scenarios. These contributions support the relevance of the LG-HOO algorithm in the context of optimization experiments and show how the algorithm can be used to support continuous optimization of online systems in stochastic scenarios.

6.1 Introduction

Traditional requirements engineering relies on domain experts and market research to model the user behavior and define requirements for their systems. However, research shows that, often as 70-90% of the time, companies can be wrong about their customer preferences [26], [38], [113]. In this scenario, several companies are adding on top their requirements engineering practices the usage of post-deployment data to evaluate the user behavior and set prioritization and optimization objectives. One way use post-deployment data in software development is through online controlled experiments with user behavior. Aligned with a set of long-term business goals metrics [114], these companies are running business-driven experiments, such as A/B tests, to validate their business hypotheses [115].

This movement started with web-facing companies such as Microsoft, Google, Facebook, Amazon, LinkedIn, among others [21], [23], [26], [28], [108], and they continuously report the competitive advantages that experimentation techniques such as A/B delivers in business-driven experiments [24]. As these companies scale their experimentation infrastructure and organization to keep their competitive edge, they developed sophisticated techniques to run experiments in range of situations that simple A/B experiments face limitations. Some of these techniques are overlapping experiments [23], optimal ramp-up [28], networked A/B testing

[22], multi-armed bandits [116], counterfactual analysis [34] and optimization experiments [33], [117]. With the increasing number of experiments being run every year [28], [38], companies are looking for new techniques that can free some of their research and development resources from the lower risk optimization experiments and allow these resources to be employed in experiments that have higher risk and higher potential return on investment and that cannot be managed automatically by a computer.

In this context, bandit algorithms started to be employed by software companies to simplify the experimentation process in some experiments [116]. Bandit problems is a class of problems that deals with the exploration/exploitation dilemma [56]. This work focuses on a subset of bandit problems called the infinitely many-armed bandit problems. This subset investigates the optimization of parameters in a continuous space, in the presence of an unknown mean reward function. This class of problems is particularly important in online controlled experiments, as several user-behavior assumptions are captured in the systems in the form of constants that can be mapped in a continuous space. The most prominent and least restrictive algorithm for the infinitely many-armed bandit problem is the Hierarchical Optimistic Optimization (HOO) algorithm [53], [118]. However, there is no evidence or empirical evaluation of the usage of this algorithm in online experiments. During the implementation of the HOO algorithm in collaboration with Sony Mobile, this algorithm presented limitations some limitations, such as the computation time, the correctness of the output based on different mean-reward distribution functions, the lack of a criterion to select the best arm at any point, and an estimation of the mean-reward.

The contribution of this chapter is three-fold. First, we provide a modification of the HOO algorithm to overcome the identified online experiments restrictions, improving the correctness of the output, the time to compute, a criterion to select the best arm and an estimation of the mean-reward function. We call this new algorithm as the Limited Growth Hierarchical Optimistic Optimization algorithm (LG-HOO). Second, we present an implementation of LG-HOO in an industrial setting, in collaboration with Sony Mobile. Third, we provide statistical evidence that supports the modification of the algorithm not only in real-world scenario, but also on simulation scenarios.

6.2 The χ-bandit problem and the infinitely many-armed bandit problem.

The χ-bandit problem, also known as the continuum-armed bandit problem, is formulated similarly to the multi-armed bandit problem. However, instead of a predefined and finite number of arms (Arm $a \in \{a_1 \ \dots \ a_K\}$), the χ-bandit problem has an infinite number of arms that are drawn from a continuous set (Arm $a \in \chi$, where $\chi \subset \mathbb{R}$). The χ-bandit problem is part of the general problem of the infinitely many-armed bandits (when the number of arms in much greater than the allowed number of plays). Some of the advantages of selecting the arms from a continuous space compared to the discrete many-arms counterpart are: (1) discretization of the space reduces limits the optimization precision to the discretization interval. To obtain a more refined interval it is necessary to add new arms that have lower confidence compared to the existing ones. (2) It is not necessary to discretize and compute the arms prior to the experiment, as well as keeping statistics for them all. (3) infinitely many-armed bandits require less exploration time than finite armed bandits (discretized) in the same conditions [119].

The χ-bandit problem can be represented as finding the arm a^* that minimizes the regret function:

(12) $$a = \pi(\delta), \text{Arm } a \in \chi, \text{where } \chi \subset \mathbb{R}$$

(13) $$y = r(a, \delta'), \text{Reward } y \in \mathbb{R}$$

(14) $$Regret(t) = r(a^*) \cdot t - \sum_{s=1}^{t} \mu(a_s)$$

$$a^* = \operatorname{argmin}_a Regret(t) \quad (15)$$

The infinitely many-armed bandit problems have been studied in different frameworks, Bayesian, frequentist parametric and frequentist non-parametric settings. The Bayesian problem is to compute the optimal actions efficiently, while the frequentist is to achieve a low rate of regret [120]. A class of algorithms for the frequentist non-parametric setting is the hierarchical optimization. A recent report compares Bayesian and the frequentist non-parametric frameworks concluding that major advantage of a hierarchical optimization algorithms is that they are faster in term of time complexity [118]. In the frequentist non-parametric framework two algorithms stand out, the Bandit Algorithm for Smooth Trees (BAST) and the Hierarchical Optimistic Optimization (HOO) [53]. In this work we use the HOO algorithm, as the BAST algorithm makes strong assumptions on the unknown mean-reward distribution functions that might not be valid in real-world applications [53]. An in-depth discussion and comparison with other algorithms are presented in [118], [120].

6.2.1 The HOO algorithm

The Hierarchical Optimistic Optimization (HOO) algorithm [120] investigates the infinite many-armed bandit problem. This algorithm is classified inside of the hierarchical optimization algorithms in the frequentist non-parametric framework. In this section, we briefly describe the HOO algorithm and its assumptions.

The algorithm makes the stochastic assumption of the mean-reward of any new selected arm. This assumption means that the reward from the new arm is an independent sample from a fixed distribution. The reward is assumed to be in the interval of $y \in [0,1]$. This assumption is realistic as the reward metric is defined and can be normalized to this range. The other assumption is that the unknown reward function is continuous around the maximum, which is a reasonable assumption in practical problems [120].

The algorithm aims to estimate the underlying unknown reward function f around its maxima while it estimates loosely f in other parts of the space χ. This is implemented using a binary tree in which each arm is associated to a region of the space. The deeper the tree grows the smaller the subset of the space χ that it estimates. The HOO uses an optimistic estimate B, using the upper confidence bound, for each node. The tree is traversed and at each iteration the node of largest bound B, is selected. Based on the reward the tree is updated.

The algorithm starts from the root and selects the child with the largest bound B (ties are broken randomly) until it reaches the leaf. From the traversed path from root to leaf, it randomly selects one node to play. The node statistics are updated and the tree is extended if it is a leaf. The statistics and bounds are computed recursively from the leaf until the root using the formulas below. Below is the notation used in throughout this work (and is the same as the one presented in [120]).

$v((H,I))$ is the value of the node (H, I).

a_{played} is the value of the played arm.

$B_{h,i}$ is the bound for the node i at the height h. The children for this node are $B_{h+1,2i-1}$ and $B_{h+1,2i}$. The root is denoted by the index (0,1)

n is the current discrete time instance and the mean reward for time is represented by $\widehat{\mu_{h,i}}(n)$.

$T_{h,i}(n)$ is the number of times a node was played until time n.

The bounds are updated according to the formulas below:

$$U_{h,i}(n) = \begin{cases} \widehat{\mu_{h,i}}(n) + \sqrt{\frac{2\ln n}{T_{h,i}(n)}} + \nu_1 \rho^h, \text{if } T_{h,i}(n) > 0 \\ +\infty, \text{if } T_{h,i}(n) = 0 \end{cases} \tag{16}$$

$$B_{h,i}(n) = \begin{cases} \min\{U_{h,i}(n), max\{B_{h+1,2i-1}(n), B_{h+1,2i}(n)\}\} \text{ if } (h,i) \in \text{Tree}_n \\ +\infty, \text{otherwise} \end{cases} \tag{17}$$

Apart from the mentioned advantages of χ-armed bandits algorithms in comparison with grid searching, one of the main advantages of this method is the updating of confidence bound of the whole path as a child is selected. Even though a particular node was not played, its confidence bound is updated if any of its descendants are played. This leads to tighter confidence intervals of a whole path. In most grid search and regular multi-armed bandits, the confidence intervals are created and updated only for the discrete played arm.

6.2.2 Related work

Optimization in online experiments can be done by using a range of different techniques. The simplest one is conducting sequential A/B/n experiments. This technique has the advantage of having comparable sample sizes for all variations in the statistical analysis at the expense of increase in the regret and the higher sample size for the optimization. Genetic algorithm has also been used in simulation of online experiments. Tamburrelli and Margara [33] proposed an infrastructure and a genetic algorithm to optimize HTML web pages in a large space. However, the proposed solution requires using non-validated assumptions on the hyper-parameters and on the mating strategies. Additionally, the solution requires a large space of unique users that makes it application in real world restricted to very large-scale software companies.

Multi-armed bandits algorithms provide a framework for optimization of experiments and it is widely used in industry [117], [121], [122]. Google's Vizier [117] is a tool for black-box optimization that take advantage of multi-armed bandit algorithms. While the paper does not focus on online controlled experiments, it mentions the use of the tool for optimization of web properties such as thumbnail sizes and color scheme. Shang [118] presents an overview of black-box optimization methods using bandits algorithms and Gaussian processes. Mattos et al. [105], [106] presents an architecture framework and architecture decisions to run optimization experiments with a domain specific heuristic for the bandit problem.

6.3 Research Process

This research was conducted in collaboration with Sony Mobile Communications in Lund, Sweden. Sony Mobile is a subsidiary of Sony Corporation and is a leading global innovator in information technology products for both consumer and professional markets. One of the Sony Mobile's products is transitioning to data-driven development and aims to run experiments continuously throughout its development process. The product consists of a business to business solution, where the user of the software consists of employees of the company that requested the solution.

The software development of this product span development for web, mobile, backend systems and distributed embedded hardware. Therefore, the requirements for an experimentation system include the ability of allowing experiments to be run in the variety of supported systems and the capability of supporting both traditional A/B experiments as well as search solutions in a larger or continuous space. An experimentation system (called ACE) that fulfills the requirements was developed following the framework and architecture decisions presented in [105], [106]. An extensive discussion of this architecture is provided in Chapter 4.

During the development of the product several assumptions were made, such as numerical, textual, and GUI constants that has a direct impact in the how the user interact with the system.

Some of the numerical assumptions are constants in the real space or in a predetermined range ($x \in \mathbb{R}$ or $x \in [0,1]$). The development team of this product wants to optimize these constants and to verify these assumptions in based on actual user behavior metrics. The HOO algorithm was selected as the starting point for the optimization search process. The HOO algorithm was implemented in the ACE system and was repeatedly tested and iterated in both simulation and with real users. The results of these iterations were constantly discussed with the product development team and the modifications of this algorithm resulted in the LG-HOO algorithm. The limitations of the HOO algorithm, the changes motivation for the LG-HOO algorithm, and an empirical comparison between both are presented in the Section 6.4.

6.4 The LG-HOO Algorithm and the Empirical Data

This section presents the modification version of the HOO algorithm [120]. The HOO algorithm was modified to allow its application in online controlled experiments. The LG-HOO follows the same structure as the HOO with the main modifications highlighted below. The growth restrictions motivate the name Limited Growth HOO. The implementation source code, the results for comparison, and the raw data used and the analysis source is available at https://github.com/davidissamattos/LG-HOO.

A node (arm) is only allowed to grow if it has been played a minimum number of times. This ensures that each arm has a minimum confidence level to ensure the growing in more confident direction. The HOO grows based only on the upper bound of the arm, and this bound can be unrealistic if only one observation has been made, as it grows with $\sqrt{(2\ln n)/T_{h,i}(n)}$. The tradeoff of selecting a minimum growth limit is that it needs a higher number of plays to reach the same level of interval refinements (that is related to the height of the tree). However, as shown later, the minimum growth does not imply that the estimated best arm is further to the theoretical best arm when the underlying function is known.

The original HOO does not point a method for selecting the best arm, as it is intended to be a continuous process. The algorithm indicates that the highest node on the tree represents the maximum of the underlying function. However, in online experiments, after a period of time the company might want to stop the experiment to save resources, improve performance or make a static decision regarding the change. Given these constraints we defined the process to select the best arm as the node (h,i) with the largest criterion $C_{h,i}$, where

$$C_{h,i} = \begin{cases} \frac{\widehat{\mu_{h,i}}(n)}{\sqrt{\frac{2\ln n}{T_{h,i}(n)}} + \nu_1 \rho^h}, \text{if } T_{h,i}(n) > 0 \\ 0, \text{if } T_{h,i}(n) = 0 \end{cases} \tag{18}$$

The idea behind this criterion is to select the node with the largest average while having the lowest bound. This penalizes nodes that have been play few times compared to nodes that have a higher confidence. A downside of this is that it favors nodes that had several plays, and this is usually associated with nodes at lower heights. However, this criterion performs better than the suggested highest height node, when measuring the distance to the theoretical best arm using an absolute Euclidian distance. Note that this criterion does not influence the growth of the tree as it still uses the upper confidence bound to select the best child node.

The LG-HOO introduces a restriction to the height of the tree. As the tree grows it becomes computationally intensive to update all bounds in a single play iteration. Delays and computational restriction if running in computers with limited resources (such as embedding the algorithm in mobile apps) can significantly impact the user experience. Restricting the tree height puts an upper bound in the computational time, however it limits the precision that the algorithm can reach.

To facilitates the understanding of the user behavior after running the algorithm for a limited time period, we make an estimation of the underlying mean-reward function using the Savitzky-Golay filter [123] with the decision criterion. Empirically, we determine that a window size of (number of nodes)/2 and a polynomial order equal to the tree's height, to produce good results. The tradeoff of using the Savitzky-Golay smoothing filter is the underestimation of high derivative peaks, leading to a conservative estimation.

In practice, it often happens that an experiment is coded and then launched without being active (showing for all users the same variation). When the experiment is finally launched several users might be using arms that are not the defined root of the HOO algorithm. Similar situation can also happen in approximation of values by different users/clients. The LG-HOO tries to minimize the number of lost data points by selecting the closest node. If the played arm is closer to the node then its children, the reward is added to the node, otherwise it is discarded. This strategy works under the assumption of continuity of the underlying function while it minimizes the number of discarded data points.

Algorithm 1 represents the full LG-HOO strategy, using the same notation as the HOO, as discussed in the background. This algorithm is implemented in Python 2.7 and is available at https://github.com/davidissamattos/LG-HOO.

Algorithm 1 The LG-HOO algorithm

Input: $\nu_1 > 0$, $\rho \in (0,1)$, minimum growth $\gamma > 0$, tree maximum height σ
Initialization: $\tau = (0,1)$ and $B_{1,2} = B_{2,2} = +\infty$

procedure SELECT ARM(n)
  $(h,i) \leftarrow (0,1)$
  $P \leftarrow (h,i)$
  while $(h,i) \in \tau$ **do**
    if $B_{h+1,2i-1} > B_{h+1,2i+1}$ **then**
      $(h,i) \leftarrow (h+1, 2i-1)$
    else
      if $B_{h+1,2i-1} < B_{h+1,2i+1}$ **then** $B_{h+1,2i-1} > B_{h+1,2i}$
      else
        Z Ber(0.5)
        $(h,i) \leftarrow (h+1, 2i-Z)$
    $P \leftarrow P \cup (h,i)$
  $(H,I) \leftarrow (h,i)$
  Selected arm $\leftarrow U(1, P_{H,I})$
procedure UPDATE TREE(a_{played}, n, reward Y)
  Select the closest arm a_{played} to the node $v((H,I))$
  if $|a_{\text{played}} - v((H,I))| < \frac{|v((H+1,2I)) - v((H+1,2I-1))|}{2}$ **then**
    $a_{\text{played}} \leftarrow v((H,I))$
  else
    break
  for all node in $P_{H,I}$ **do**
    $T_{h,i} \leftarrow T_{h,i} + 1$
    $\hat{\mu_{h,i}} \leftarrow (1 - \frac{1}{T_{h,i}})\hat{\mu}_{h,i} + \frac{Y}{T_{h,i}}$
  for all node in τ **do**
    $U_{h,i} \leftarrow \hat{\mu}_{h,i} + \sqrt{(2\ln n)/T_{h,i}} + \nu_1 \rho^h$
  if $T_{H,I} > \gamma$ **and** $H < \sigma$ **then**
    $B_{H+1,2I-1} \leftarrow +\infty$
    $B_{H+1,2I} \leftarrow +\infty$
  while node $\neq (0,)$ **do**
    $(h,i) \leftarrow$ new leaf
    $B_{h,i} \leftarrow \min\{U_{h,i}, \max(B_{h+1,2i-1}, B_{h+1,2i})\}$
procedure SELECT BEST ARM
  $a_{\text{best}} = \max_\tau (\frac{\hat{\mu}_{h,i}}{\sqrt{(2\ln n)/T_{h,i}} + \nu_1 \rho^h})$
procedure APPROXIMATION OF THE UNDERLYING FUNCTION
  $\hat{f} = \texttt{Savitzky-Golay}((\frac{\hat{\mu}_{h,i}}{\sqrt{(2\ln n)/T_{h,i}} + \nu_1 \rho^h}), \text{number of nodes}/2, \max_\tau h)$

The repository presents additional information on the connection of the algorithm with the implemented code. The algorithm is composed of four procedures. The first procedure is the procedure that selects the arm to be played in with the current tree. The second is called after an arm is played and a reward is received, updating and extending the tree. The third selects the best arm, when the optimization process is being finalized. The forth estimates the mean-reward function using the Savitzky-Golay filter.

6.4.1 The LG-HOO in simulation

In this subsection, we provide some illustrative pictures of the LG-HOO algorithm in a simulation environment using different mean reward functions. Figure 6.1 shows the usage of the LG-HOO algorithm in 6 different conditions. The orange line is the true mean-reward function that determines the probability of a Bernoulli distribution $Y = Ber(f(x))$. Where Y is the measure value (0 or 1, click or no click) and $f(x)$ is the mean reward function with variation x. This line can represent a customer profile (that is unknown but we still want to optimize a variation for this function). This profile can be complex as the picture in the left-top corner or simpler such as the picture in left-middle with only three ranges of value. The optimization process consists of finding the variation x that maximizes the mean reward function based only on the stochastic measured Y.

These simulations show how the LG-HOO algorithm work and estimate the mean reward function (blue line). All the simulations were conducted considering a total of 10,000 unique interactions (horizon $n = 10{,}000$), using the minimum growth of 10, maximum tree height limit of 10, $\nu_1 = 1.0$, and $\rho = 0.5$, which is representative of the amount of data collected in a period of one month of the conducted experiment with Sony Mobile. We can see that with this number of unique interactions we can estimate the parameter that maximizes mean reward function.

6.4.2 The LG-HOO at Sony Mobile

The LG-HOO was implemented in the context of the product described in Section 6.3. One of the features of the product has an algorithm that estimates the time for launching a notification to users. If the notification arrives too early the users can ignore it and the feature has little value. If it arrives too late it can have a negative impact in the overall user experience. Before the experiment, the feature was using the minimum time scenario (reducing even more the time makes the notification arrives too late). The impact of the notification is measured depending on the action the user takes after receiving the notification. This metric is a stochastic variable that follows a Bernoulli distribution, where 1 (positive value) represents when the user takes an action in time and 0 when the user does not take an action in time (negative). The metric is stochastic because different factors not related to the time of the notification might influence the user action. The team wanted to investigate if a change in the algorithm that modifies the notification time impacts the metric. The hypothesis of this experiment is that adding a constant delay in the algorithm could indicate the extent the algorithm influences the metric and if development effort was needed to improve it. The team also wanted to minimize the regret of too early notifications. Sequential A/B/n experiments would take too long to cover the whole extent of search space while increasing the regret. This scenario sets an appropriate experiment for a continuum-armed bandit algorithm such as the LG-HOO.

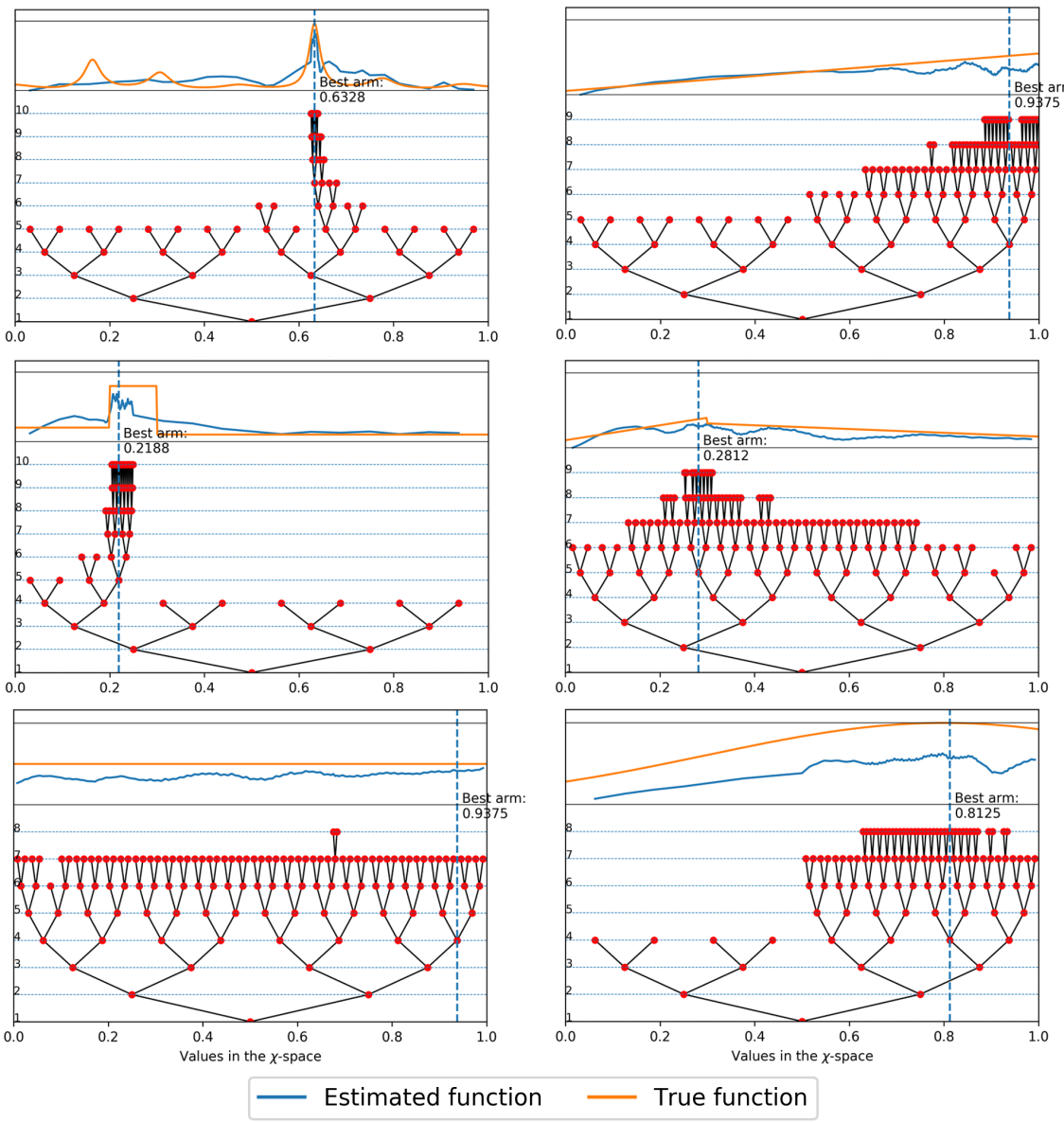

Figure 6.1 - Simulation results of the LG-HOO algorithm in wide range of different user mean-reward functions. In orange, is the true mean-reward function (unknown to the LG-HOO). In blue, is the estimated mean-reward function. The tree represents the LG-HOO search process at the end of the iteration, and the blue vertical line represents the best arm selection using the proposed selection criterion. The top-left subplot represents the same mean-reward function discussed in the original HOO algorithm [120].

The experiment consisted of searching an appropriate delay offset for the notification. The experiment limited the offset in the range of 0 and 600,000 milliseconds (10 minutes). The users were assigned to a new variation delay every time they launched their mobile applications, and they logged their behavior right after the timeout to complete the action.

The LG-HOO was implemented in the ACE system in Python 2.7. The ACE system is hosted in the Google App Engine Flexible cloud environment[4]. The company application logged data and requested variation arms from the ACE system using POST requests. In case of lost packages or failure in requesting a new variation, the system uses the current variation (offset of zero). The parameters of the LG-HOO in this scenario are: minimum growth of 10, maximum tree height limit of 10, $\nu_1 = 1.0$, and $\rho = 0.5$. The limit in the tree height restricts the precision of the output of the algorithm in approximately 500ms, which is considered good level of

[4] https://cloud.google.com/appengine/docs/flexible/

precision for the application. For this experiment, it was collected data from over 5000 user interactions in the period of 4 weeks. The results and the outputs of the algorithm are shown in Figure 6.2. This Figure provides both the visualization of the search tree, as well as the approximated mean-reward function and the selected best arm. The mean-reward function indicates that the offset does not have a large influence in the selected metric for the extent of the whole range of delays, but it still shows that a small delay can improve the concerned metric.

For the team the approximation of the of the mean reward function was important because it maps how the users behave in respect to this modification on the system, and therefore can help decisions such as to modify the feature, try other experiments on the feature or related features or move the development effort to another part of the system.

This section described the LG-HOO algorithm, the modifications, and trade-offs of the LG-HOO. In the simulation subsection, we provide simulation results and evidence of the LG-HOO being applied to different mean-reward functions. The simulation results allowed us to implement the LG-HOO algorithm with confidence in an industrial setting in collaboration with Sony Mobile. As there is no industrial evidence of the use of the HOO algorithm, some of its limitations were unknown prior to this work. The industrial case provides real-world evidence of the use of the LG-HOO in online experiments.

6.5 Discussion

Prior to launching the algorithm to real users, a comparison between the HOO and the LG-HOO was made and is discussed in this section. The algorithms were compared using the absolute Euclidian distance to the theoretical maximum and the time to compute an algorithm iteration. The first comparison looks at how far the algorithm got from the true value and relates to the following LG-HOO modifications: (1) the selection of the best arm policy modification and (2) the minimum number of times an arm must be played before growing. The second comparison

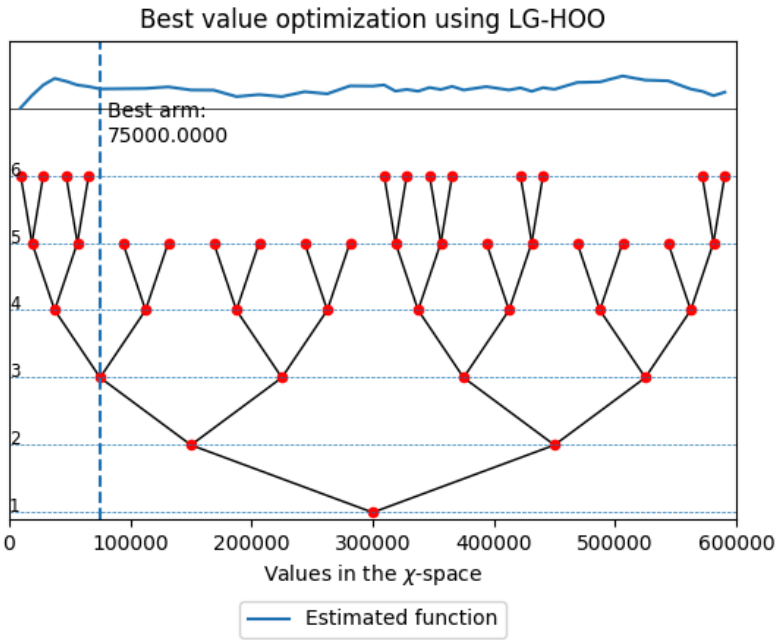

Figure 6.2 - The LG-HOO used in the Sony Mobile case. This picture provides both the visualization of the search tree, as well as the approximated mean-reward function and the selected best arm.

Table 6.1 - Summary of the statistical analysis to compare the LG-HOO and the HOO algorithms using the Mann-Whitney U test, using a confidence level of 95%

Metric	Algorithm	Mean value	Absolute relative difference	p-value
Euclidian Distance	LG-HOO	0.293	14.3%	0.04179
	HOO	0.335		
Time spent (in seconds)	LG-HOO	1.00	26%	< 2.2e-16
	HOO	1.26		

relates to degradation of user experience and performance of the system due to the introduction of delays in the estimation of the next arm to be played.

The algorithms were compared using a Monte Carlo simulation comparing one thousand runs with a horizon of $n = 1000$. At each simulation of the algorithm, it was used a generated random polynomial function as the true mean-reward function $f(x)$. The polynomial functions were generated by: (1) generating a set of 30 random points in the (x, y) plane, (2) fitting a polynomial with random order (ranging between 0 and 10) to these points, and (3) constraining both the space (x) and the mean reward probability (y) between 0 and 1. The user follows a Bernoulli distribution $Bern(f(x))$, where $y = f(x)$, and 1 represents a success. With this method we generate random polynomial functions that are used as the mean reward functions to simulate the user profile for the algorithms. With this method we can simulated both algorithms against the same set of true mean reward functions and compare the LG-HOO and the HOO solutions with the true solution using the absolute Euclidean distance.

The time spent in the calculation for selecting the next arm and the Euclidian distance were done in the same hardware and operational conditions. The data collected from this Monte Carlo simulation, and the conducted analysis is also available at repository. The collected data for the Euclidian distance and the time spent metrics for both algorithms are non-normal, Shapiro-Wilk test with $p < 2.2e\text{-}16$ and by visual inspection. Therefore, we compared the two algorithms metrics using the Mann-Whitney U non-parametric test [124]. We considered as null hypothesis that the respective LG-HOO metric does not differ from the HOO metric.

Table 1 provides a summary of the statistical analysis. This statistical analysis provide evidence that the LG-HOO reduces the distance average Euclidian in 14.3% and reduces the spent time in 26%, using a confidence level of 95%. Due to the increased performance of the LG-HOO regarding to the correctness of the output and the computation time, only the LG-HOO algorithm was selected for empirical evaluation in the company case. The data and code to run this statistical analysis is available at the repository.

6.6 Conclusion

Optimization procedures associated with bandit algorithms are of great interest to companies running online experiments. A particular case is the optimization of a continuous space in the presence of an unknown mean-reward function. As companies develop their products, several user assumptions are incorporated into constants in their software development. Optimization in this scenario is a subclass of bandit problems called infinitely many-armed bandits. Previous research provides algorithms to solve this problem in the unidimensional space. However, these algorithms do not have empirical evidence or usage in online experiments and have restrictions that prevent their utilization as proposed. This work explores the unidimensional infinitely many-armed bandits problem in collaboration with Sony Mobile Communications.

The contribution of this work is three-fold. First, we present a modification of the Hierarchical Optimistic Optimization algorithm (HOO), called the Limited Growth Hierarchical Optimistic Optimization algorithm (LG-HOO). This modification is intended to overcome the problems associate with implementing the HOO algorithm in real-world online experiments. The modifications and the trade-offs involved with these modifications are presented. Second, the LG-HOO was implemented in collaboration with Sony Mobile. In this scenario, we provide real-world evidence of the usage of this algorithm for optimization of software constants. Third, we provide a statistical comparison between the LG-HOO and the HOO algorithm in simulation. The statistical analysis supports the conclusion that the LG-HOO perform better than the HOO, in the time spent to run and the accuracy of the results. These contributions support the relevance of the LG-HOO algorithm in the context of optimization experiments and show how the algorithm can be used to support continuous optimization of online systems in stochastic scenarios.

This work is the first step in analyzing the usage of infinitely many-armed bandit algorithms in optimization procedures in software development. In future work, we plan to expand the LG-HOO to support multi-dimensional arm space, support a multi-dimensional reward, as these are one of the key aspects that companies want to provide optimization, and validate these extensions in relevant industrial problems.

7 Pitfalls in multi-armed bandits for online experiments

This chapter is based on the following publication:

Multi-armed bandits in the Wild: Common Pitfalls in Online Experiments
David Issa Mattos, Jan Bosch and Helena Holmström Olsson
In submission to an international software engineering journal, 2018.

Chapter Summary
One of the techniques to continuously validate and deliver value in online software systems is the use of controlled experiments, also known as A/B tests. Although controlled experiments can be used to evaluate incremental changes and fine-tune system parameters, these applications are repetitive and can create a significant impact on the overall cost of experimentation. The class of reinforcement learning algorithms called Multi-Armed Bandit (MAB) algorithms appears to be a technique that can overcome traditional A/B experiments, by delivering faster results with a better allocation of resources. However, the incorrect use of MAB algorithms can lead to misinterpretations and wrong conclusions that can potentially damage the company's business. The objective of this chapter is to analyze and understand the pitfalls and restrictions of using MABs in online experiments, and what strategies are used to overcome them. We used a multiple case study with five leading software companies in online experiments together with simulations to analyze the pitfalls and restrictions. Additionally, we provide strategies for practitioners to avoid and overcome such pitfalls a guideline for to select an experimentation technique that minimizes the occurrence of these restrictions and pitfalls.

Although multi-armed bandits can provide a better allocation of resources in online experiments, its implementation in practice is associated with several restrictions and pitfalls. We aim for the results of this work to be used as a reference material for practitioners to avoid pitfalls while implementing MAB applications for online experiments.

7.1 Introduction

Delivering software that has value to customers is a primary concern of every software company. Previous research has shown that, often, the prioritization of software features in the development process is driven by beliefs, past experiences and the role of power inside the organization, and these are not necessarily aligned with customers' needs [6], [9]. The decision to invest development resources and engineering efforts on features that do not have a confirmed value can result in losses and opportunity costs to the software companies [5].

Web-facing software companies (such as Microsoft, Google, Netflix, Booking.com, Yelp, and Amazon, among others) often report success cases and the competitive advantage of using post-deployment data together with online controlled experiments as an integral part of their development methodologies [6], [21]–[24], [26], [28], [116], [125]. This competitive advantage leads companies to start experimenting in almost every change made in their systems, from developing new functionality to the fine tuning and optimization of existing functionality, resulting in thousands of experiments being deployed every year [25], [28], [38]. One example of the extension on how experiments are being used to optimize systems is the '50 shades of blue' experiment at Google. In this experiment, Google's engineers ran an experiment to

determine the best shade of blue for a hyperlink in Google's search page [126], [127]. The best shade of blue resulted in an additional 200 million dollars in revenue.

The use of experiments to fine-tune and to evaluate incremental changes in existing features requires, in general, a higher number of users, due to the small effect sizes between the different variations. Fine-tuning experiments are repetitive and the cost of running several small tuning experiments can create a significant impact on the overall cost of experimentation in the organization. To support such a scale in the number of experiments while keeping the costs of running experiments low, software companies and academic researchers are developing innovative solutions in automating and scaling the experimentation infrastructure [23], [28], [38], [114], and in developing and adopting new algorithms to run experiments [34], [53].

The online experimentation problem can be phrased as a Multi-Armed Bandit (MAB) problem that examines the exploration and exploitation trade-off. MAB (also known as K-armed bandit) problems are a class of problems classified inside the umbrella of reinforcement learning, that sequentially experiment with the goal of producing the greatest reward [47], [56]. In online experiments, determining which variation generates the best value to the user is the exploration part, and deploying the best solution to the highest number of users (increasing the value delivered to users) is the exploitation part. MAB algorithms are commonly compared to A/B testing and they promise to deliver faster results with a better allocation of resources [34]. However, the incorrect use of MAB algorithms can lead to misinterpretations and wrong conclusions that can potentially hurt the company's business. Through a multiple case study with software industries conducting online experimentation, this chapter identifies and explores the restrictions and pitfalls of MAB algorithms such as: (1) naïve implementation of MAB algorithms, (2) MAB algorithms that violate the assumptions, (3) usage of MAB algorithms in experimentation frameworks that do not support them, and (4) the different goals when choosing between MAB and A/B testing.

MAB algorithms have been studied for over 50 years and have been successfully implemented in online experiments and many different domains. However, to the best of the authors' knowledge, there is no work that discusses the restrictions and pitfalls in the implementation of MABs for online experiments or which provides strategies or guidelines to avoid such pitfalls. This work addresses this gap from the industry perspective by looking at practitioners' experience and discusses how to solve or avoid the identified restrictions and pitfalls. We utilize a combination of a multiple case study method with simulations to triangulate the data. This research goal is captured in the following research questions:

RQ1: What are the restrictions and pitfalls associated with MAB algorithms applied to online experiments?

RQ2: What are the solutions and strategies used by companies to overcome these restrictions and pitfalls?

This research provides guidance to practitioners when choosing experimentation strategies for online experiments with three main contributions. First, to the best of the authors' knowledge, this is the first work to analyze the restrictions and pitfalls of MAB algorithms applied to online experiments. Second, this work provides strategies for practitioners to avoid and overcome such pitfalls. Third, this work provides a guideline for practitioners to select an experimentation technique that minimizes the occurrence of these pitfalls.

The remainder of the chapter is organized as follows. Section 7.2 discusses the research method. Section 7.3 presents and discusses the restrictions and common pitfalls associated with MAB implementations for online experiments. Section 7.4 presents a discussion of the results, use cases where MAB algorithms are desired and a guideline process to select between traditional

experimentation techniques such as A/B experiments and MABs. Section 7.5 discusses the validity threats of this research. Section 7.6 concludes and discusses related research challenges.

7.2 Research Method

Using earlier research with several online companies, we identified that, although academic research suggests that MAB algorithms provided several benefits compared to A/B experiments, these companies did not apply these algorithms in practice. Some of these companies suggested that these algorithms did not provide benefits over the traditional A/B testing in their use cases, due to the high complexity of the experimental design and the presence of some restrictions and pitfalls that either minimized the benefits of these algorithms or invalidated the experiment results. This work studies the restrictions and pitfalls of MAB algorithms identified by companies when implementing and deploying these algorithms in their systems, rather than the mathematical analysis and simulation results usually provided in academic research. We also discuss strategies adopted by companies on how to avoid such pitfalls and achieve trustworthy results from the experiment.

To answer the two research questions proposed in the introduction (Section 7.1), we used, as a research methodology, a multiple case study following the design and process described in [71] triangulated with simulations. The multiple case study allowed us to explore why different companies are, or are not, using these algorithms and to understand the rationale behind the decision of not using MAB algorithms despite the apparent advantages. The simulations allowed us to verify situations mentioned by the interviewees in the case study and, when possible, triangulate the data.

7.2.1 Multiple case study

The case study process was conducted in four stages: (1) Definition and planning, (2) Data selection and collection, (3) Data analysis, and (4) Case study report. An overview of each of the four stages will now be provided.

7.2.1.1 Definition and planning

Based on the observations from earlier research that despite the benefits suggested by research, online companies were not using MAB algorithms, we designed a multiple case study to understand: (1) the restrictions and pitfalls that companies running online experiments encounter when using MAB algorithms (RQ1), and (2) how these companies overcome these restrictions and pitfalls (RQ2).

7.2.1.2 Data collection and analysis

The main data-gathering instrument used in this multiple case study was thematic semi-structured interviews based on an interview guide. Our choice for thematic semi-structured interviews was based on the flexibility of asking open questions to discuss a broad range of problems inside the same theme [71].

The case study companies and the subjects were selected based on the criteria that the companies run online experiments on a daily basis or develop solutions for online experiments, including A/B testing, factorial experiments, and/or MAB algorithms. All the subjects selected for the case study had experience of online experiments and knowledge of either developing or running MAB algorithms in their systems. This selection criteria limits the interviewees to a restrictive pool of experts available outside academia. This is reflected in the position occupied by the interviewees, where almost all are senior or principal data scientists and developers.

This multiple case study was conducted together with five companies between the period of August 2017 and January 2018. The primary data sources of this study are transcripts of audio recordings, notes of the meetings, email communication, and information shared by the

interviewees, such as slides and technical reports. A total of 11 interviews were conducted. All interviews had a minimum duration of 33 minutes and a maximum duration of 70 minutes, with a mean value of 40 minutes and a median of 38 minutes. The number of interviews was decided based on the saturation criterion, where no new information or points of view were gained from new subjects [71]. The interviewees' selection represents a high number of experts with knowledge and experience in both online experiments and MABs.

The interviews consisted of a questionnaire containing four general open-ended questions aimed at identifying problems using MAB algorithms seen in practice, identifying the restrictions and pitfalls of these algorithms that prevented their utilization in practice, identifying good use cases where these algorithms can be used, and identifying the strategies used by the companies to overcome these limitations. The interview started with a brief introduction and description of the research, such as the context in which we are interested (online experiments) along with our goals. Following this, we asked the participants about their familiarity with MABs and A/B testing, and their industrial experience with them. In the context of online experiments, we asked if they had experienced any pitfalls with MABs and cases where the algorithm provided results that were difficult to explain or contradictory to other techniques such as A/B testing. We asked about identified restrictions that prevented the practitioners from using these algorithms. These restrictions could be technical, business, or organizational restrictions, such as: development cost and need of more experts in this area; lack of support by current experimentation architecture; or lack of a business case to implement the algorithms. Next, we asked how they overcame these pitfalls and these restrictions, and what would be good cases for MABs in the context of online experimentation. We also reported, while keeping the companies anonymous, the restrictions and pitfalls identified by the other companies, to see if similar pitfalls or restrictions were also present. During the coding and the data analysis, any information that was not clear or that needed further explanation was discussed again with the interviewees.

7.2.1.3 The companies

Due to reasons of confidentiality, we have provided only a short description of the companies and their domain. Table 7.1 provide a list of the interviewers' position in each company

Company A is a multinational conglomerate company that manufactures consumer electronics and provides software solutions for consumers, professionals, and business-to-business solutions. Different teams inside this company are running A/B/n experiments and have conducted the preliminary research and implementation of MAB algorithms for experimentation in their software products. In company A, we interviewed four practitioners working with two different products.

Company B is a multinational technology company that develops, manufactures, and sells software solutions and services ranging from operating systems to web solutions. Several of the company's products run A/B/n experiments on a regular basis or at scale, and they have also successfully implemented and run MAB in some of their products and have evaluated MAB algorithms for their online experimentation platform. In company B, we interviewed a total of four practitioners working in the same experimentation team but with different products.

Company C is a company that develops experimentation solutions for its customers. The company offers A/B/n, MVT and other experimentation tools for websites along with frameworks for experimentation in mobile platforms. The company developed their own statistics engine and offers solutions using MAB algorithms to customers. The company's customers include several multinational companies from different domains, from software companies, to entertainment, to large news agencies. We interviewed two practitioners involved with the MAB solutions.

Company D is a software company focused on website optimization and offering experimentation tools and solutions for A/B testing and MABs. One practitioner was interviewed, the director of data science in the company. However, since the interview this interviewee has left the company and provides consultancy in the area of experimentation and MAB algorithms, with experience in developing A/B experimentation platforms and MABs for several multinational companies, from software companies to large news agencies.

Company E is a travel fare aggregator and travel engine provider. It develops booking and travel solutions used by both individuals and the travel industry. A/B testing methodologies are an integral part of the development process of the company. The company is introducing and evaluating MAB algorithms in their online experimentation process. One practitioner was interviewed. The interviewee is a senior data scientist and is responsible for the development and evaluation of a MAB tool to be used in internal development.

The interviews were conducted either in person on the company's premises, or online via video conference. In addition to the interviews, some of the reported restrictions and pitfalls by practitioners are accompanied by simulation studies comparing the involved techniques. These simulations were suggested during the interviews and were conducted after the interviews' data collection. The simulations represent additional evidence to the related restriction or pitfall identified in the interviews.

7.2.1.4 Data analysis

The analysis of the interview data was performed in two steps. First, the empirical data for each interview was transcribed where applicable and thematic coding was applied [128]. From the coding, we identified the restrictions and pitfalls as well as the potential solutions. The second step consisted of grouping the coded restrictions, pitfalls and potential solutions. Practitioners reported restrictions and solution strategies through different examples and in the context of their work. However, some of the restrictions, pitfalls and solution strategies could be grouped into a more general problem statement. This grouping procedure was analyzed and interpreted by all authors.

Table 7.1 - Overview of the position of each interviewee

Company	Position
A	Senior developer
	Senior developer
	Master architect
B	Principal data scientist
	Principal data scientist
	Principal data scientist
	Data scientist
C	Product manager
	Senior statistician
D	Director of data science
E	Senior data scientist

7.2.2 Simulations

During the data collection of the multiple case study, some of the interviewees also shared simulation designs (simulations 7.3.1.2 and 7.3.3.2) to illustrate some of the concepts. These

simulations were replicated and presented in this work. We also provide an additional simulation (simulation 7.3.6.2) that were not discussed during interview times, but that verify the interviewers' comments on the pitfall. The objective of these simulations is to triangulate the data shared by the interviews with hypothetical scenarios that could happen when using multi-armed bandits. Each simulation is presented as a hypothetical scenario and it is presented and discussed in details in the respective pitfall section.

7.3 Results

This section discusses the results obtained from the collected empirical data from the interviews and from the simulations.

7.3.1 Decision errors in naïve MAB implementations

Traditional controlled experiments follow traditional statistical analyses and aim to minimize and control type I errors, namely the incorrect decision of rejecting the null hypothesis when the null hypothesis is true. For experiments with features, type I errors refer to deciding to invest or ship a feature that is not providing any value to the system. Traditional A/B experimenting statistical analyses tend to focus on controlling for type I errors, while minimizing type II errors by increasing the predetermined sample size and duration of the experiment.

7.3.1.1 Pitfall

An identified pitfall is a naïve implementation of MAB in experiments that require the control of type I errors. The naïve implementation selects the best ranked arm as the desired decision without taking into account confidence intervals or controlling for type I errors. This is often the output of MAB algorithms in research [53]. Although this strategy works for minimizing regret and error type II, it can lead to several mistakes that can impact the company's decision-making process. This pitfall is usually observed in the first few MAB experiments and in the first iterations of the algorithms in the experimentation tool. This pitfall occurs when companies introduce MAB algorithms without in-depth knowledge, or without explicit goals for the output of the algorithm, or when using third-party experimentation tools that implement naïve algorithms. This pitfall was identified by company A in third-party systems and E in first-time implementations of MAB algorithms. Company E also identify this pitfall when adapting and using MAB-based recommendation/content-serving systems in online experiments applications. According to company E, one of the reasons this pitfall persists is because it is common that academic MAB publications are concerned with other aspects of this class of problems and do not mention or implement statistical analysis on top of MAB algorithms.

"*We already made a few decisions based on the result of this third-party tool when we decided to compare it with another A/B testing tool. The results were so different that we decided to investigate the reason*" – Senior Developer Company A

7.3.1.2 Simulation

To illustrate this scenario, we simulate a simple decision-making process based on a traditional A/B experiment and three naïve implementations MAB algorithms (ϵ-greedy 10%, Softmax 10% and UCB1). Suppose an online company decides to use a third-party experimentation service that provides both A/B and MAB algorithms. The feature that is going to be experimented, does not influence the metric. However, adopting this feature increases the time spent by a team maintaining and developing the feature. The service provides statistical analysis and confidence intervals for the A/B experiments (based on a χ^2 test using a significance level of 95% [6]), as several competitors also offer. The experimentation service advertises MAB as the next level for online experiments and considers it a big differentiator as most competitors do not have it. However, for MAB only a ranking of the best arms is provided. This simulation

investigates type I decision errors in three different sample sizes in a similar situation, where there are no difference between the variants.

The conditions of the simulation are: each algorithm was simulated with a Monte Carlo (simulated 1000 times) procedure for three different sample sizes. The first horizon has 2000 unique samples representing a power of less than 60%. The second horizon has 4000 unique samples, representing a power of 80%. The third horizon has 8000 unique samples, representing a power of over 95%. All the sample size calculations are calculated based on the necessary power to conduct an A/B experiment with a significance level of 5%, initial conversion baseline of 10%, and absolute effect size of 2% [6]. The confidence intervals were obtained by utilizing a Bootstrap resampling (with n=1000) procedure in the simulation results.

$$A, B \sim Bernoulli(0.1)$$

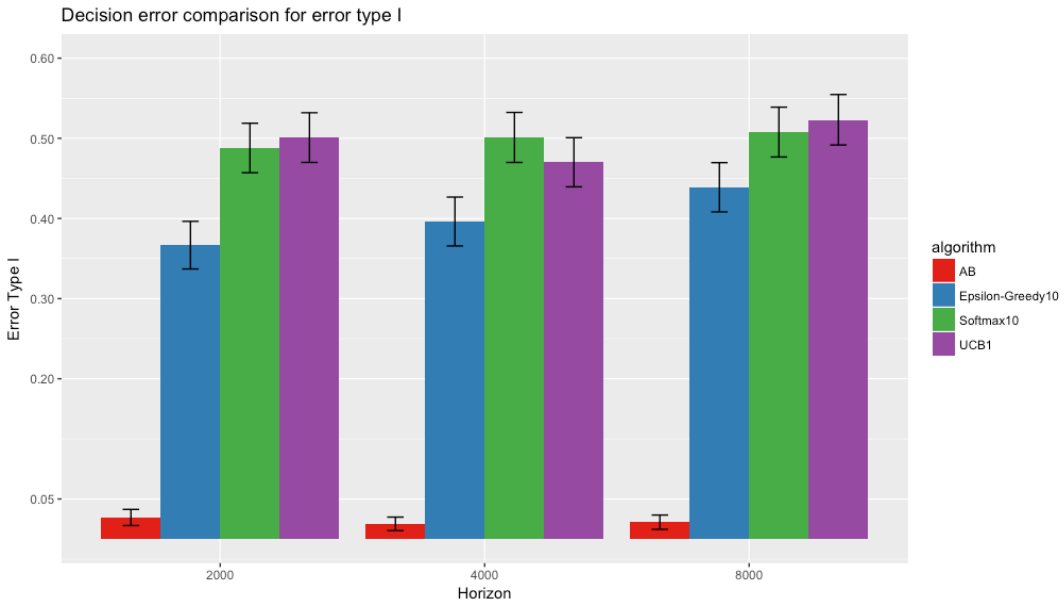

Figure 7.1 - Comparison of the decision error for type I error in A/B experiments (red) and naïve implementations of MABs. Algorithms that have variable exploration rates have a higher decision error.

Figure 1 shows the percentage of error type I in the Monte-Carlo simulation. It can be seen in this picture that the A/B experiment process still makes error type I, but this error is constrained to the pre-defined significance level of 5%. However, the MAB algorithms often make error type I. If MAB algorithms are used in such situations, the experiment has a negative impact in the company, as many features that were incorrectly validated now are maintained and developed by the company, but without any real impact in the metrics.

7.3.1.3 Strategies

Experiences from companies A, B and E suggest that the experimentation organization and the experiment owner should first estimate the consequences of committing each type of error and then clearly identify how the outcome of the experiment will be used in the future development and evolution of the system. The aim of this analysis is to select strategies that fulfill the goal of the experiment and avoid potential decision-making errors.

If the estimated cost of a type I error is high, a good strategy is falling back to the traditional A/B experiment scenario, which already provides consistent processes and tools. In this case, type I errors can negatively impact the system in the future in both performance and maintenance. If the estimated cost of a type I error is low and the aim of the experiment is to minimize regret or type II errors, practitioners can use naïve implementations of MAB

algorithms. This scenario is closer to experiments for optimization, for example tuning parameters. If the goal of the experiment is to both minimize regret and control for type I errors, it is necessary to implement a more rigorous statistical analysis on top of the MAB algorithm. In this scenario, the algorithm is still minimizing regret during its execution, but the decision-making process also follows traditional methods. The disadvantage of this method is the lack of balance between the variants. This lack of balance can limit the statistical method for the analysis and reduce the statistical power of the experiment. This implies a longer experimentation time.

Some tools available for experimentation use MABs as the default algorithm (e.g. in Google Analytics [129]). For those systems, a careful analysis of the implementation (if available), the assumptions of the experiment, and the expected outcome of the experiment will be required. If possible, the results from MAB algorithms should be analyzed with their respective confidence intervals as this can provide a more in-depth comparison between the different arms and generate additional insights.

7.3.2 Bad variation lockdown

One of the requirements of most online experiments is to keep user experience consistent throughout the experiment's duration [6], [21]. In A/B testing, users are assigned to different variants and companies aim to keep the user experience consistent (if the user is exposed to a feature, the user should have continuous access to this feature during the experiment). Several methods can be used to keep the experience consistent. Common methods include: using deterministic randomization reassignment [6], [21], requesting a variation only once and caching it in the application (such as in cookies); cross-checking and comparing the variant during every assignment. Deterministic randomization reassignment is traditionally computed through pseudo random number generation and hashing functions (such as SHA1) on the unit of randomization (e.g. cookies, user ID, instance number). This procedure ensures that every time the system requests a variation, the assignment system always replies with the same variation, guaranteeing that the user is constantly exposed to the same variant during the entire experiment duration. After the experiment is completed and a decision about the variants is made, the users can be assigned to a fixed variation (one of the treatments or the control variation) until the next deployment. The caching and the cross-checking methods have drawbacks compared to the deterministic randomization reassignment, such as being difficult to scale in both the number of simultaneous experiments and number of users, lack of support for ramp-up and automated shut-down [6].

7.3.2.1 Restriction

In A/B experiments, the user's experience consistency depends on the experiment definition (e.g. choice of the randomization unit) and how the experiment affects the user experience. Implementations of MAB should also take into account the user experience consistency. However, several MAB algorithms do not support deterministic randomization reassignment in their current formulation and implementation, as they do not use randomization units in their assignment process. Therefore, several MAB algorithms rely on cross-checking and caching methods, which introduces complexity, limits the scaling, and restricts the use of ramp-up and automated shut-down. If a user is assigned to a clearly bad variant (or arm), will this user be locked to this variant alone until this experiment is shut down in a new deployment? Such a situation could potentially hurt a small percentage of the users for a long period, if workarounds in the caching and cross-checking mechanisms are not implemented. This problem was identified by companies A and B.

Additionally, company C reports a challenging in keeping user consistency in MAB algorithms in long-term experiments (such as continuous optimization experiments) with limited and

recurring users. Long-term experiments can run for between months and years and are often associated with content serving experiments. In this scenario, the experiment has a limited number of users that are exposed to the same variant multiple times. The users are asymmetrically assigned towards one variation, which might not be the optimal one, as the optimal variation is under-sampled. This situation prevents both the minimization of the regret and limits the statistical power in future statistical analysis, therefore threatening the validity of the experiment and the confidence in the experiment results. If implementing an extra reassignment procedure, how often and in what ways are users reassigned? If the reassignment occurs seldomly, the validity of the experiment and the regret minimization are at stake. If the users are reassigned very frequently, this could create a user experience inconsistency. Other alternatives such as only reassigning the users of a lower variant could influence the exploration rate and possibly the regret minimization, if not analyzed carefully.

"*If user consistency is necessary we strongly encourage our customers to go with A/B testing route instead of bandits*" – Senior Statistician Company C

7.3.2.2 Strategies

Successful implementations require a careful study and understanding of the implications of user consistency and when it is necessary. If user consistency is mandatory, a reasonable alternative is to use traditional A/B experiments, where tools and techniques for keeping this consistency while maintaining ramp-up and automated shut-down are available with deterministic randomization reassignment. If using caching or cross-checking methods, it is necessary to implement an extra layer to the application code in order to handle ramp-up and automated shut-down cases, increasing the complexity of the experiment. However, if such a layer is already present, user consistency can be implemented when using MAB algorithms.

Long-term MABs should not be used for exploration purposes or for understanding of the system and the user behavior. In such cases, having control of type I errors and predetermined power are of greater importance than regret minimization. If the goal of the experimentation procedure is to understand both the system and the users, the A/B experimentation procedure can provide better insights without reassignment and variant lock restriction. If the aim of the experiment is to conduct a long-term optimization procedure, a short-term A/B experiment can be executed first. This experiment aims to understand the functionality/feature under experiment and how the unit of diversion performs. What is the randomization unit? How often do recurring randomization units request a new variant? To what extent should user experience be consistent for this experiment? Do users perceive a significant experience change between variants? Those questions can be answered with an A/B experiment prior to the long-term MAB optimization. Content-serving MAB applications (such as serving ads and news) often do not impact the user experience negatively but optimizing layouts and other user interface elements might pose problems that need to be carefully analyzed.

7.3.3 Decision errors due to violations of assumptions

MABs are often compared to A/B experiments and are applied to the same type of problems. A common pitfall is to assume that the A/B testing assumptions are the same and are valid for MAB algorithms. Most MAB algorithms assume that the reward observations are identically and independently distributed.

7.3.3.1 Pitfalls

A common violation is when there are changes in the distribution over time. An example from websites is how the user profile changes with the days of the week and month [6]. In the case of ad clicks, the distribution of clicks could be:

$$X(t) \sim \begin{cases} Bernoulli(p_1), \text{ if } t = \text{weekdays} \\ Bernoulli(p_2), \text{ if } t = \text{weekends} \end{cases}$$

This might occur with because, in weekdays, the system has more business activity, while in weekends it has more home activity. Similar situations happen in ecommerce, where the click distribution changes depending on the day of the month. Due to the concurrent exploration and exploitation, sample allocation is not constant and the MAB can learn and allocate exploitation to a temporary arm. Depending on the algorithm, it can take a long time to unlearn this temporary arm. Therefore, questions such as "should I launch my experiment in a Tuesday or on a Saturday" [130] and seasonality effects [64] arise. In this case, MAB should take into consideration a longer time compared to A/B experiments to compensate for these effects (but also having the risk of being affected by another cycle).

A variation of this assumption violation is the novelty effect, which also changes the distribution in time [17], [21]. When adding a new feature to the software, the novelty effect of this new feature might increase a metric (such as usage) in the beginning but with a significant drop of usage in time. After a long period, the metric of the system with the new feature might have a statistically significant lower metric compared to the original system.

Other assumptions violations that can happen in multi-armed bandits are:

(1) the correlation between different arms. Algorithms such as UCB1 are not appropriated to understand and exploit correlations in the arms. The usage algorithms that make independence assumptions with correlated arms might lead to an overestimation of the best arms as well as a weak exploitation of the solution space [55].

(2) the presence of delayed reward feedback (compared to the experiment horizon) [131]. While this is true for click events, some other metrics commonly tracked in A/B experiments have dependency on time, e.g. 3 days' or 15+ days' user retention. If MABs are running with those metrics, the arm selection might be suboptimal and can even lead to wrong conclusions. The pitfall is using time-dependent metrics which have a delay is not negligible compared to the horizon as the reward for the MAB experiment.

(3) the re-scaling and normalization of metrics. Some MAB algorithms, e.g. UCB1, require the reward to be constrained between 0 and 1. Therefore, existing metrics (that do not comply with this restriction) need to be re-scaled and normalized to fit into this constraint. However, the re-scaling and normalization transformations need to be carefully analyzed, implemented, and validated before applying to a MAB algorithm. Common pitfalls in this process that can change are: 1) Transformation on the fly, all new data or group of data points change the transformation rule, e.g. subtracting current expected value or dividing the result by the current standard deviation. As the experiment progresses, the metric changes and cannot be compared. 2) Utilizing historical data that are no longer valid, e.g. a transformation procedure that relies on data that are not representative anymore. 3) Carrying transformations from previous experiments and metrics to new ones, e.g. using 3 days' user retention transformation for a 15+ days' user retention metric might result in an incorrect measurement

7.3.3.2 Simulation

The assumption violation with the novelty effect is explored in the following simulation. Suppose a company is launching a new voice in the voice assistant tool for their mobile application. Prior to launch the company post videos of the feature, make advertisements creating some user expectation. Consider that the variant A is the current voice and B is the new voice. The company is measuring if the user uses or not the feature during the day. The goal of this simulation is to show that even with traditional statistical analysis on top of the algorithms, MAB can still inflate error type I if the assumptions of the algorithms are not met.

For this simulation, we consider the following distribution for the usage of the voice assistant tool. Where 1 represents that the user has used the tool and 0 represents that the user has not used the voice assistant tool during the day. t is a discrete positive variable that represents the day after launch ($t = 0$ means the launching day, $t = 10$ represents the feature usage after 10 days). For the variant B, we consider that it starts with a higher usage due to the novelty effect but it slowly declines until it reaches a constant usage level of 3% lower than the baseline variant A. This situation violates the assumptions of most MAB algorithms, but does not violate the assumptions of an A/B experiment. The distribution of the response for each variation can be represented as:

$$A \sim Bernoulli(0.1)$$

$$B(t) \sim \begin{cases} Bernoulli(0.12 - 0.005t), \text{for } t < 10 \\ Bernoulli(0.07), \text{if } t \geq 10 \end{cases}$$

The simulation runs for a total of 20 days. Each day has the same number of users. Therefore, the variant B performs poorly more than half the time. In this simulation, we are interested in the number of times the system makes an incorrect decision in selecting alternative B (making a type I error). To compensate for the decision errors' effects of naïve MAB decision-making (as discussed in section 7.3.1), this simulation implements the same statistical analysis and decision criteria for all algorithms (A/B and the MAB algorithms). The decision is based on a

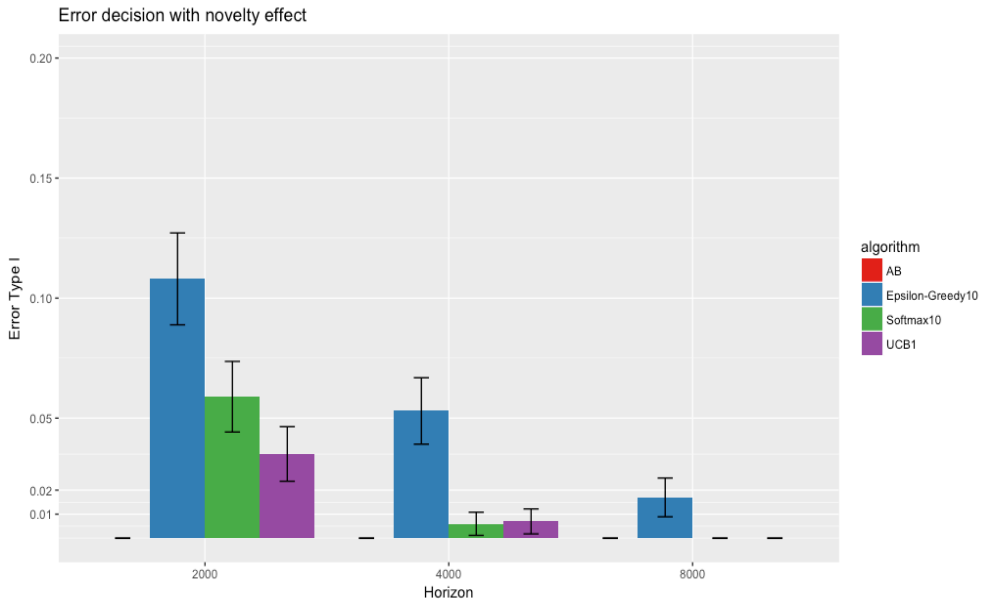

Figure 7.2- Comparison of the decision error for type I error, with a simulated novelty effect. For MAB, the errors decrease as the experiments run for longer (higher horizon size). However, even with the same statistical decision-making framework of the traditional A/B experiments, MAB algorithms do not have the same robustness as A/B experiments with rewards distribution changing with time. In the picture above, there were no A/B experiments' decision errors

final collected data and the process detailed in [6] using a significance level of 95% (or $\alpha = 5\%$).

The conditions of the simulation are: each algorithm was simulated with a Monte Carlo (simulated a 1000 times) procedure with for three different sample sizes. The first horizon has 2000 unique samples, representing a power of less than 60%. The second horizon has 4000 unique samples, representing a power of 80%. The third horizon has 8000 unique samples, representing a power of over 95%. All the sample size calculations are calculated based on the

necessary power to conduct an A/B experiment with significance level of 5%, initial conversion baseline of 10% and absolute effect size of 2% [6]. The confidence intervals were obtained by utilizing a Bootstrap resampling (with n=1000) procedure in the simulation results.

Figure 2 shows that shows that in the presence of non-identically and independently distributed users and arms MAB algorithms can inflate error type I ($\alpha = 5\%$), and this is proportional to the power of the experiment. In underpowered experiments, MAB algorithms with the same decision-making process of A/B experiments are not able to control for type I errors and have a significantly higher incidence of incorrect decisions.

7.3.3.3 Strategies

To overcome time-invariant assumption violations, different strategies can be used. The first strategy is to choose a re-weighting scheme for the MAB. Most algorithms use a constant average scheme that places equal importance to all samples. Different schemes, such as moving averages or exponential averages can be used to shift the reward importance with time [55]. A second strategy is to use contextual bandit algorithms. These algorithms take into account domain-specific knowledge to provide and personalize the exploration space. However, these strategies add complexity to the algorithm in both implementation and the number of hyper-parameters to tune. Moreover, these extensions require specific domain knowledge of the system and the user behavior with time. Such knowledge might not be available at the time of the experiment or might require a pre-A/B experiment to test some of the assumptions.

"*You need to be careful when deciding to go for bandits, if you know that some of these assumptions were violated you need to do a proper statistical modelling of the problem and validating it before deploying your algorithm*" – Director of Data Science, Company D

In the delayed reward assumption, different solutions for the MAB with delayed observations were proposed, based on different assumptions. However, it is still a computationally difficult problem [131]. Another solution is to run long-term MAB algorithms in order to minimize the effects of the delays. However, modifying metrics and adding context information to MAB algorithms also require validation.

"*Research commonly takes the bandit reward and context for granted, but the process of selecting a good context and reward is iterative and slow. In practice, it is very difficult to compare two bandits and how they perform against each other if you are using different contexts*" – Senior data scientist, Company E

For decision-making processes and the evaluation of new features in cases where you have uncertainties about the problem assumptions, the traditional A/B experimentation process provides a more robust framework to evaluate the results.

7.3.4 Lack of Sample Ratio Mismatch quality check in MAB algorithms

Sample Ratio Mismatch (SRM) is considered to be one of the critical diagnosis tests for A/B experiments [132]. This test allows checking if the percentage allocation of users for each variant is within an expected confidence interval and validate the randomization system.

Suppose that a company is running an A/B experiment where 50% of the users are allocated to each variation. Variation A is assigned to 99,000 users and variation B is assigned to 101,000. Comparing this user distribution with the designed proportions of 50% for each using a χ^2 test indicates that the current user distributions is very unlikely to be in the 50% designed proportions ($p < 0.01\%$). Detecting SRM early helps to prevent randomization bias towards any variation and allows the experimentation organization to check performance and instrumentation issues. If one variation has a worse timing performance difference (because of caching issues, for example) compared to the other, the results might be biased. Sample Ratio

Mismatch is commonly conducted in the pre-experiment A/A test phase [6] and during the experiment execution as a guardrail metric [39].

7.3.4.1 Pitfall

MAB algorithms base the regret minimization in asymmetric sampling favoring one of the variations, by design it is not possible to run SRM checks. Due to this characteristic, it is hard to identify randomization and instrumentation bias in the system during the experiment execution.

"*We don't deny the advantages of MABs for recommendation systems, but if we can't run quality checks we don't trust our data and then we don't have a business case for using it*" – Principal Data Scientist, Company B

Without proper validation prior to the experiment execution, MABs can be minimizing the regret towards a biased variation. This pitfall was mainly identified with Company B. However, companies A, C and E also confirm the difficulty in implementing MAB quality checks and validating and guaranteeing that the MAB implementation is not being biased due to external factors (to the algorithm itself) such as the instrumentation and randomization system.

7.3.4.2 Strategy

A strategy employed by company A is to evaluate the MAB experimentation system (instrumentation, metrics, randomization techniques, etc.) using traditional A/A experiments, and even A/B/n experiments in a reduced arm space. A/A experiments allow the experiment organization to not only test run sample ration mismatch, but also other quality checks to validate the instrumentation and randomization system before moving towards a MAB. However, the MAB algorithm implementation is not validated. A/B/n experiments in a reduced arm space allow the organization to determine the best arm and then proceed to checking MAB in this limited arm space. If the MAB confirms the results of the A/B/n experiment, the experimentation organization can proceed with more confidence to a MAB algorithm in the full arm space. The biggest disadvantage of this strategy is the time and resources taken to validate a MAB experiment prior to its launch, therefore restricting the solution to long-term optimization processes, where the return on investment for the MAB is higher than outcomes produced by A/B/n strategies, or to parts of the system when the company already trusts the collected data and the algorithm was already validated in previous experiments. Company C provide MAB algorithms for their customers, however guarantying the correctness of the algorithms only against multiple test and use cases in their framework.

7.3.5 Increased complexity in ramp-up procedures in MAB algorithms

Ramp-up is a risk-minimization technique [6], where the treatment variations are only launched to a small percentage of the population. Large effect sizes require a smaller sample size to detect the metric movement with the same statistical power. Moreover, large variations in key metrics are typically related to experiment errors [108]. Ramp-up allows the experiment organization to try experiments impacting upon a small percentage of users if a large degradation in key metrics is present. If the experiment does not present any severe degradation, the percentage of users that are exposed to the treatment is increased until the variations have equal allocation to maximize the power of the experiment [6].

7.3.5.1 Pitfall

Although most MAB algorithms will allocate a very small slot to bad variations, they will also allocate a large slot to metrics that have a large positive movement. Large positive movements in metrics are rare [108] and can indicate instrumentation problems. Because of this adaptive allocation, algorithms require careful study if combined with ramp-up procedures.

MAB ramp-up solutions can be divided in reset solutions, which reassign the users to a new solution at every step at the ramp-up, and consistency ramp-up, where the initial assignment is maintained at each ramp-up step. One problem that can arise with ramp-up solutions that do not reset in each variation is the presence of bias. If the initial allocated percentage is not representative, the MAB can learn and converge to a biased arm. When ramping-up, the system might take a long time to unlearn this solution while exploring a suboptimal arm.

7.3.5.2 Strategy

We present three approaches to combine MAB with ramp-up. The first alternative is the use of algorithms that permit adjusting the exploration step; these would include ϵ-greedy variations. Variations of this algorithm allow setting the exploration step at a significantly higher level than the exploitation, reducing the adaptive allocation of the arms. This solution can be combined with ramp-up procedures, where, at every increment in the percentage of users exposed, the exploration rate is also reduced. This approach has the disadvantage of reducing the regret minimization

A second approach is to provide a hard allocation boundary for all treatment variations. This approach should be carefully designed in such a way that it does not degrade the exploration and the decision process as the percentage of users allocated to the experiment increases.

The third approach is creating a hard reset for each ramp-up procedure. This will reset the combination of best arms and allow the system to re-explore. This approach has the disadvantage of increasing time to converge to a solution and might create reassignment problems (that impact upon the user experience), as previously discussed.

7.3.6 Increasing regret in experiments due to Simpson's Paradox in MAB algorithms

A different problem connected to the asymmetric allocation of users present in MAB is the Simpson's Paradox [34], [133]. The Simpson's Paradox (SP) refers to the observation that "*a statistical relationship observed in a population – i.e. a collection of subgroups or individuals – could be reversed within all of the subgroups that make up that population*" [133]. Kievit et al. [133] provide several examples of Simpson's Paradox observed in experimental in the areas of social science, medicine and biology. Different research discuss SP in online experiments [34], [39], [65]. This example is adapted from Crook et al. [65].

Suppose an experiment on a website text is implemented in more than one country (UK and Sweden). The experiment consists of reducing a long sentence instruction to a short direct instruction. This experiment is launched in the two countries with the same sample size (determined by the power calculations). Supposing that the power calculation indicates that the UK would run with 10% of users assigned to the treatment and in Sweden it requires 50% in the treatment group. It is possible that the treatment variant performs better in both countries but they both countries are combined the control variant performs better. However, note that the SP can occur just in one or all subgroups. Kievit et al. [133] provide a guide to identify SP and discuss ways to minimize the occurrence of SP. However, there is no single mathematical property that SP instances have in common. The most common mistake is to assume that findings at the level of group generalizes to subgroups, or to individuals over time.

7.3.6.1 Pitfall

In the example, a confounding variable (the country) that was not taken into account in the randomization process makes the randomization process not randomly distributed. In the (unknown) presence of the SP together with the adaptive allocation of the MAB algorithm reinforce the selection of the worst variant increasing the regret of the algorithms. Although the regret is also increase in other experiments, the identification of the sampling bias conditions

[39], [65] is aggravated by the non-randomly distributed form that MAB allocate the users to the variations.

This pitfall was identified by Company B, although they report that they haven't made an in-depth analysis, as MAB is not the core part of their experimentation strategies. Company A discussed the presence of SP as a problem seen in their A/B experiments. As some their MAB applications are aimed in known situations after A/B experiments, SP can be identified in stage prior to the implementation of MAB. However, they recognize that the presence of SP would invalidate the goal of minimizing regret in MAB applications.

7.3.6.2 Simulation

We present simulation that shows the increase of the regret of multi-armed bandits in the presence of a SP condition. This simulation is based on the hypothetical SP case presented by Lihoung Li in [134]. Suppose we are recommending two news articles {sports, movies} to users {male, female}. The confounding factor (could be due to an algorithm bias for example) is that male users receive more sport news articles (75%) than female users (25%), and female users receive more movies news articles (75%) than male users (25%). Table 7.2 below shows the click-through rate for each subgroup and the general group. From the table we can see that the sports perform better than movies for both male and female, but movies perform better if the results are aggregated. The SP case is compared to the case (called no-SP) where the selection between male and female are not confounded with the group sports or movies. In this case the sports group will have an overall CTR of 0.6, and the movies group will have an overall CTR of 0.5.

Both the SP and no-SP cases are simulated for the algorithms UCB1 and Epsilon-Greedy 10%. Figure 7.3 shows the cumulative regret for each algorithm in each case (with SP and no-SP) with a horizon of 10,000.

This simulation shows that the presence of unknown SP in multi-armed bandits can lead to a significant increase in the regret and therefore minimizing the advantage of this kind of algorithm in this situation. The cumulative regret grows linear with the horizon, but at different

Table 7.2 – SP case for the simulation

CTR per group	Male	Female	Overall
Sports	0.4	0.8	(0.4x0.75+0.8x0.25) = 0.5
Movies	0.3	0.7	(0.3x0.25+0.7x0.75) = 0.6

rates depending in presence or absence of SP. This suggests that introducing multi-armed bandits in long-term experiments in the presence of SP can lead to unexpected outcomes in terms of the predicted regret bounds.

7.3.6.3 Strategy

The identification of SP is a not trivial as there are several types of SP and there are no shared property between them [133]. The foremost recommendation by Kievit et al. [133], is not to assume that relationships at the group level also hold for subgroups or individuals over time. This analysis requires incorporating in research designs data collection to facilitate the

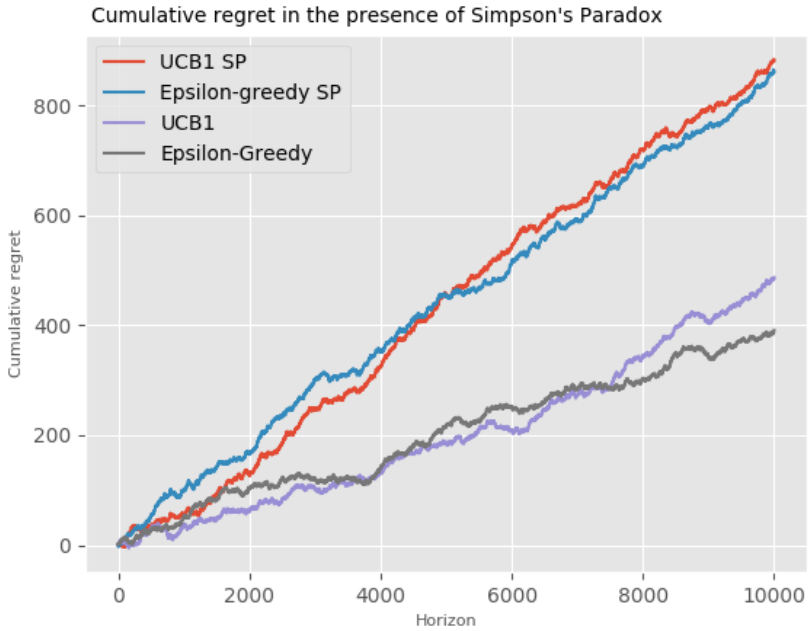

Figure 7.3 – Comparison of the cumulative regret in MAB algorithms in the presence of SP and without SP. In the picture lines red and blue indicates the cases where SP is present and lines purple and brown indicates the cases without SP.

comparison of patterns across the different group levels. Crook et al. [65] illustrate some situations where SP were identified in online controlled experiments and give a caution against of combining metrics over type, especially when the proportions of the control and treatments vary, or when the subpopulations are sampled at different rates. Traditional experiments such as A/B/n and factorial designs offer a better framework for identifying SP compared to MAB, A/A test and sample ratio mismatch tests for each segment group, A/B experiments and segmentation of data collection prior to the MAB experiment can provide the necessary insights to identify SP.

However, if the SP was already identified several alternatives can be taken by the experiment organization. Common strategies are eliminating the confounding variable (if it is a sampling error bug), choice of different statistical tests, eliminating the data that generates the SP (i.e. when SP appears in ramp-up situations). If the SP is intrinsic to the problem, the confounding variable is identified and minimization of regret is still desired the experiment organization can introduce an exogenous contextual variable (representing the confounding variable) in a contextual bandit algorithm [52].

7.3.7 Adaptive allocation based on a single metric

Most optimization by experiments algorithms and applications focus on the optimization of a single metric, such as CTR, revenue or retention. However, this scenario proves to be limiting to organizations that utilize a wide range of metrics to reach a decision. The decision-making process of online experiments involves the analysis and trade-off of several important metrics, from both feature-level metrics to product-level metrics. Well-developed experimentation organizations use a composed success metric, the Overall Evaluation Criteria [26], guardrail and data quality metrics in their decision. Before exploiting, the decision process may involve other stakeholders in order to understand the implications of these metrics in the system.

7.3.7.1 Restrictions

Due to the adaptive allocation of arms in MAB algorithms, a solution is already exploited based on the single success metric. This prevents the usage of other supporting metrics for an in-depth analysis. This can bias the algorithms towards an arm in a way that is not totally comprehensible and that can hurt the company's business or future goals. Moreover, incorporating additional metrics and a traditional statistical analysis on top of the MAB can be compromised because of the lack of power in some arms.

Suppose a company was evaluating multiple new layouts for the home screen of a news website using a MAB approach. Each new variation layout could be a combination of different grids varying the number of rows and columns. If only a composed metric such as overall customer satisfaction is being used, the decision of the best arm could exclude the analysis of other important metrics. For example, the new variant customer satisfaction might favor free users instead of subscribers. Some negative impacts might not be seen during the experiment duration, but it was identified in previous experiments by company and was set as guardrails metrics. As an example, the new variant might significantly increase loading time in mobile users. Initially users might attribute this to their connection, but if this persists for a time longer than the experiment duration and competitors are providing faster loading time, mobile users can move to new services.

Although these restrictions apply also to traditional experiments if other metrics are not being measure or used in the decision process, MAB incorporate the decision to exploit an arm during the execution time. Therefore, potential bad variations might be overly exploited while hurting metrics that could be used in traditional analysis.

This restriction was identified by all companies. Company C and D recommend avoiding bandits if this restriction applies to their customers experiment. The difficulty of incorporating other metrics in the exploitation decision is one of the reasons that prevents Company B of applying MAB in more general experiments. Some quotes from this case study reflect this:

"*Despite the hype with bandits, in our workflow they can only be used in very narrow cases where we know very well what we want to optimize.*" – Principal Data Scientist Company B

"*...if more than one metric is present, and our customers can't combine and generate a single metric for the bandit, we recommend them to avoid this feature at all*" – Product Manager Company C

7.3.7.2 Strategy

Some strategies allow MAB algorithms to be used with multiple metrics. The first strategy is the use of MAB extensions for multi-objective optimization with Pareto relations. However, such solutions add technical complexity and multiple failure modes to an experiment that is not present in A/B experiments. Providing a validated Pareto curve to a MAB algorithm requires a deep understanding of the system that is only achieved several experiments are conducted. This solution is discussed mainly in academic setting [135] and none of the companies reported using this solution.

The strategy is to use the current Overall Evaluation Criteria that is being used in A/B experiments. However, this metric must have sufficient sensitivity and be a leading indicator [24], so it does not fall in the case of the delayed reward pitfall described in section 7.3.3. Company C recommends this approach to their clients in specific experiments. However, in this case it is recommended to run the MAB algorithm in a smaller fraction of the user base and use the winning arm in an A/B experiment against the current variation to check against all other metrics

A third approach suggested by Company D, is to add the additional information from other metrics to a contextual bandit algorithms as exogenous variables. However, this solution has some drawbacks. The first is to have a good understanding of the problem as this solution is not exploratory. The second is that the external context also needs validation so it does create bias towards a particular context.

7.4 Discussion

Online experiments powered by MABs can provide a competitive edge to experimentation organizations if applied carefully. However, the collected empirical data suggest that the benefits of using MABs in online experiments can be hindered by several pitfalls and restrictions.

One of the reasons that MAB applications to online experiments present several of these applications is the underlying experimentation process in which the MAB algorithm is applied. The case study companies adopt experimentation models similar to the HYPEX [46], the RIGHT [3], and the Framework for Online Controlled Experiments [41]. These models make different assumptions compared to the needs of MAB applications. For example, the presented models assume that the goal of the decision-making process is to minimize type I errors (and, with sufficient users, minimize type II errors) instead of minimizing regret or minimizing type II errors at the expense of type I errors (in the case of limited number of users.). Other differences are: the iteration frequency, the focus of the experiment (innovation, learning or optimization), type of data collection (qualitative or quantitative data) and data selection (convenience sampling, stratified sampling, fully randomized or adaptive sampling).

The utilization of experimentation process models that focus on innovation and learning and use A/B experiments can lead to pitfalls such as violation of assumptions, and usage of MAB in explorative problems. The usage of techniques tailored for A/B experiments (such as ramp-up and keeping user consistency) in MAB algorithms can lead to complex designs that can increase the costs and the practical significance necessary to implement the MAB algorithms.

Besides the identified pitfalls and restrictions MAB applications face other issues that still need to be addressed for a successful implementation of this technique: organization training and system complexity. One of the key challenges in online controlled experiments is the adoption a culture of experimentation and generating trustworthy results that can be understood and used by non-experts [76]. Stakeholders consuming results from experiments need a basic understanding of the assumptions, execution, and limitations of experiments, as well as familiarity with basic statistical inference. A/B experiments is currently more widespread and understood inside several organizations and it is currently part of the toolbox of managers, developers and product owners. Companies continuously invest in the capacitating their employees to use and understand traditional controlled experiments. For example, company B provides training in A/B experiments for the company's employees at least once a month, in addition to tailored training for specific product teams. However, MAB algorithms, their assumptions and their limitations are mostly restricted to groups of experts. To be fully adopted inside the organization MAB needs to be understood and trusted by non-machine learning experts as A/B experiments are. This would require basic understanding of machine learning, MABs, the problems they address, the assumptions and how the results should be interpreted.

"*One challenges I have seen in bandits is: how do we communicate the results to other people? If we want to adopt it, we need to train our organization to understand not only MABs but also machine learning in general*" – Senior data scientist Company E

An additional problem to MAB algorithms and their implementation in production increases the complexity of the experimentation system. Practical problems to consider are the

implementation constraints and how these algorithms scale when the number of users increase. Modifications in MAB algorithms to accommodate implementation constraints can change assumptions and regret boundaries and might reflect in different outcomes. One point to

Table 7.3 - Summary of the restrictions and pitfalls

Pitfalls/Restrictions	Reasons	Strategies
Increase in type I error	• Naïve implementations of MABs	• Adding a statistical framework on top of the MAB implementation • Usage of the MAB in applications where the cost of making a type I error is low • A/B experiments
	• Assumption violations (reward distribution, delayed reward, normalization)	• Contextual bandits, reweighting scheme, delayed reward bandits, long-term optimization. These strategies require a prior understanding of the problem and the user behavior. • A/B experiments
	• Using MABs in exploration problems	• Design of experiments: A/B/n, full factorial, or fractional factorial experiments
Detecting experimentation problems	• Lack of sample ratio mismatch in MABs	• Prior A/A experiment • A/B/n experiment in a reduced arm space
	• Increased regret in MAB in the presence of Simpson Paradox	• Prior A/A and A/B/n experiments • If identified, contextual bandits can be used
	• Complexity in ramp-up procedures	• MAB algorithms with adjustable exploration rates • Hard allocation boundaries • Hard reset of the algorithm for each ramp-up iteration
Increased design complexity	• Adaptive allocation based in a single metric	• Composed Pareto relations between metrics • Sensitive Overall Evaluation Criteria (OEC)
	• Need to keep user consistency in long-term experiments or in experiments with a high number of users	• Prior A/B/n experiment • Variation reassignment procedures
	• Communicating experiment results	• Capacitating the organization in machine learning and MABs • Providing a similar presentation layer as A/B experiment results

consider is how fast the implementation is and how costly it is to develop such an infrastructure compared to A/B experiments. Previous industry research [136], [137] shows that time delays can be very costly. If the MAB algorithm is significantly slowing down the system, its implementation might be introducing costs higher than the delivered value. This scenario increases the practical significance of MAB to be adopted. Another implementation issue when introducing machine learning algorithms in production is in increasing technical debt in the system [138]. Sculley et al. [77] specifically discuss analysis debt in bandit algorithms. Analysis debt refers to the difficulty to predict the behavior of a system before it is released. In several direct feedback loop problems (where the algorithm influences the selection of its own future training data), it is common practice to use standard supervised algorithms instead of bandit algorithms. This occurs because bandit algorithms do not necessarily scale well to the size of action spaces required for real-world problems. Additionally, the maintenance and evolution of MAB systems have higher costs involved than traditional A/B experiments, due to their complexity and scalability issues.

Table 7.3 shows a summary of the restrictions, pitfalls and strategies discussed in this research.

Even though this research shows several pitfalls and restrictions, MAB still have several benefits and use cases that can lead to potential benefits and value to the experimentation organization.

7.4.1 Use cases for multi-armed bandits

Despite, the different pitfalls and restriction when using MAB algorithms in online experiments, these algorithms still provide several benefits in appropriated cases. Multi-armed bandit algorithms base their strategies in the regret minimization criteria and present several advantages in situations where this property is essential. Next, we discuss online experimentation situations where regret minimization is desired and MAB were applied successfully. The following cases represent different situations where the use of MAB algorithms is desired compared to traditional experiments. These cases were provided and discussed by the case study companies and summarized in the groups below.

7.4.1.1 Content-serving systems

Content-serving systems are designed to select and display content to users. The goals of this type of system is to provide to the users the most relevant content from a pool of possibilities. In this type of system, type II error are worse than type I. Examples of this type of systems are ad-serving systems and news systems. Multi-armed bandits provide a more integrated framework for this type of system, compared to A/B experiments. As an example, suppose a news website has space for a large headline. The website wants to increase the number of people clicking in larger headline. As new stories are written and added, the website wants to select the headlines that will provide the best conversion for the large headline. This case represents a case that is more suitable for a MAB than A/B testing. A/B/n testing does not minimize the regret as it takes time to explore the solution space and require an extensive manual labor as this system is supposed to run for a long-term. Different MAB algorithms (such as a moving-window MAB) can provide a long-term solution to this problem, exploiting the arm with the highest return, while exploring arms that can have a higher potential reward. Another example is provided by Wang et al. [139], that discuss the usage of multi-armed bandits in a content-serving music system.

7.4.1.2 Short-term campaigns

Short campaigns refer to a limited time presentation of the variant. After the campaign time, the software returns to the original variation. An example of a short-term campaign is a Christmas advertisement. A retailer website creates four versions of an advertisement for a Christmas sale. The advertisement can only be displayed after Thanksgiving and before

Christmas. In this case, the goal of the campaign is to maximize the reward (or minimize the regret). Although an A/B/n experiment can used, by the time that enough data is collected and a decision is made about the best variant, there wouldn't be enough exploitation time. As in the content-serving, MAB algorithms present a solution that can maximize the cumulative reward of the short-term campaign. Féraud and Urvoy [140] present a case study of MAB used in short-term emailing campaign.

7.4.1.3 Targeting experiments

In several cases, the experimentation organization already understands the different preferences and segmentation of their user base. Using this prior-knowledge, the experiment organization can develop custom variations to their user segments and run targeting experiments. This type of experiment is not supported in the traditional A/B/n and factorial experimentation frameworks. However, contextual multi-armed bandits provide an extensive framework to incorporate multiple context information in the experiment. An example of contextual bandits in targeting experiments in Yahoo News is described in [52], [141].

7.4.1.4 Other cases

If the previously discussed restrictions and pitfalls are addressed, MAB can be used in online experiments to achieve a faster decision regarding the best arm when the effect size and the difference in mean-reward is considered high. The algorithms used in this situation are based on variation of ϵ-greedy algorithms, to ensure that the other arms will still have been explored enough for a proper comparison. A known implementation for this case is available in Google Analytics [47], [129].

Another use of MAB is in controlling false discovery rate in sequential A/B experiments [29], [142]. In this case, a sequence of A/B tests are replaced by a sequence of best-arm MAB instances. This allows the complexity to stay relatively low, with high power experiments and low false-discovery rate.

7.4.2 Guidelines

The selection of the appropriated experimentation process and technique can avoid most of the pitfalls discussed in this research. Therefore, to aid practitioners in selecting an appropriate technique for their online experimentations, in Table 7.4 we provide an inductive recommendation guide based on the discussions and strategies of this research. This guide is composed of five broad questions regarding the experimentation planning. If, in any of the questions, the selected decision includes the traditional experimentation techniques, such as A/B experiments or factorial experiments, as described in Section 2.1, it is recommended that the practitioners utilize these techniques instead of MAB algorithms.

7.5 Validity Threats

7.5.1 Construct validity

This multiple case study analyzes the restrictions and pitfalls from applying MAB in industry applications. This study was designed with semi-structured interviews, where the open questions allowed the interviewees to express their opinions and elaborate more in the problems they faced with MABs. In all cases, we ensured that we were using a consistent vocabulary with the technical terms with which the participants were familiar. In cases where there was a difference in vocabulary terms, we had other technical synonyms and examples from the available literature and textbooks in both online controlled experiments and in MAB areas. Due to the restrictive selection process of the interviewees, both the interviewers and the interviewees used precise technical terms to avoid ambiguity.

Table 7.4 - Guidelines to select and experimentation techniques

Question	Decision
What is the goal of the experiment?	• Learning: A/B/n, full factorial and fractional experiments • Innovation: A/B experiments • Optimization: A/B/n experiments, sequential A/B experiments and MABs
What is the cost of making type I and type II errors?	• If both types of error are costly: high power traditional A/B/n and full factorial experiments • If only type I error is high: traditional A/B/n, full factorial experiments • If only type II error is high: MABs
How well known are the problem and the assumptions?	• If not well known: traditional A/B/n and full factorial experiments • If the system needs validation: A/B/n experiments with pre-quality tests • If well known: analysis if it matches the assumptions of different MABs o If the context is well understood and validated: contextual bandits

Is there a single decision metric?	• If there is only one metric
	  o And this metric is sufficiently sensitive, MABs can be used
	  o Delayed metrics and less sensitive metrics: A/B/n experiments
	• If there are multiple metrics that cannot be grouped: A/B/n experiments

How long will the experiment run?	• Short-term experiments
	  o If there are external deadlines regardless of sample size: MABs
	  o Otherwise: traditional A/B/n and full factorial experiments
	• Long-term experiments:
	  o If it is to collect sufficient sample size: A/B experiments
	  o If there is no user-consistency requirement: MABs
	  o If it is to adaptively change the variation with time: contextual bandits and non-stationary bandits

7.5.2 External validity

We identified two issues that can limit the generalization of the identified results for a large number of companies: (1) the representativeness of the studied case companies, and (2) the number of practitioners who participated in the interviews.

Regarding the first issue, in the domain of experimentation and usage of MABs, few companies actually have experience with both techniques. This can be assessed by the publications of companies in both fields. This study was conducted with five different companies working in different domains. Of these, three of them developed experimentation systems for their own products while two developed and commercialized experimentation systems for other companies. Additionally, all the interviewed companies have a global presence. The selection of the companies offers a broad view of the restrictions and pitfalls identified by medium- to large-sized companies in developing MAB algorithms for online experimentation in web systems. These results cannot be generalized for companies working in other domains and for smaller sized companies. However, online software experimentation is not a common practice in non-web facing companies such as embedded systems and in offline software products.

Although smaller sized software companies are also running online experiments they rely in software experimentation tools and expertise provided by companies such as company C and company D. MABs are not common practice in small companies if these techniques are not provided within the experimentation tools. Therefore, since the research questions address the combination of two specific techniques, we believe this study covers relevant companies with experience in both.

Regarding the second issue, we conducted this multiple case study with highly qualified people in both fields until we reached saturation [71]. We acknowledge that some other restrictions or pitfalls might not have been identified due to two main reasons: (1) the selected group of interviewees have not experienced other restrictions or pitfalls in their application domain. To address this problem, we conducted interviews with practitioners working on different products and in different companies; (2) The case companies did not want to share some pitfalls or restrictions publicly, even if anonymized. This might be related to their business strategy, as some of the case study companies commercialize products that make use of MAB algorithms.

7.5.3 Internal validity

To minimize internal validity threats [71], we took the following precautions: (1) during interviews we discussed potential restriction problems and pitfalls either identified by other companies or by traditional online experimentation methods such as A/B testing. This procedure was applied after the interviewees had the opportunity to explore pitfalls and restrictions without the bias of different companies or groups. Upon questioning, specific restrictions and pitfalls allowed the interviewees to consider and reason if they faced similar issues. The interviewees were selected by their expertise and experience in the area. (2) In the interviewees' selection criteria, we excluded inexperienced practitioners, or practitioners that were not familiar with both techniques (A/B experiments and MABs), therefore eliminating pitfalls confounded with the lack of experience in the area. However, in the context of experimentation teams, such pitfalls would be avoided if inexperienced developers joined an experienced team. (3) At the end of all interviews we created a summary of the discussed pitfalls, restrictions and strategies, giving the opportunity for the participant to rephrase and further discuss any of the pitfalls or restrictions. The discussion of other related restrictions and pitfalls also helped the company to confirm and further discuss the pitfalls and strategies. The final discussion was important to clarify all the discussed aspects and remove interpretation ambiguities. (4) Some of the companies did not permit us to record of the interviews so the researchers had to rely on their interview notes. Immediately after each interview, the researchers discussed and complemented their interview notes with a self-contained summary of the discussion of each restriction, pitfall and strategy. During the grouping of the coded restrictions and pitfalls, in one interview an identified pitfall conflicted with pitfalls reported by other companies. In this case, clarification for this pitfall was made with a follow-up telephone call.

7.6 Conclusion

Delivering software that has value to customers is a primary concern of every software company, and A/B experiments are one of the prominent techniques used by web-facing software companies. In this context, MAB algorithms were developed and suggest that they may be capable of delivering faster results with a better allocation of resources. This work presents both the common models and paradigms of applying online experiments and the MAB framework together with three common algorithms. In a multiple case study with five leading software companies in the area of online experiments, we interviewed 11 experts in both online controlled experiments and MAB algorithms. This study identified nine restrictions and pitfalls of using MAB algorithms in online experiments and discussed strategies to overcome them.

These results are used to propose a guideline to aid practitioners in selecting an appropriate technique for their online experimentations. This work contributes in three main aspects. First, to the best of the authors' knowledge, this is the first work to analyze the restrictions and pitfalls of MAB algorithms applied to online experiments. Second, this work provides strategies for practitioners to avoid and overcome such pitfalls. Third, this work provides a guideline for practitioners to select an experimentation technique that minimizes the occurrence of these pitfalls.

The application of MAB algorithms for online experimentation is a new topic that has not been deeply explored. As more companies and researchers start to explore this area, new restrictions, pitfalls and strategies will be identified. In future work, we plan to explore the pitfalls and strategies of MAB algorithms in recommender systems and how these strategies can be applied in the context of online controlled experiments. Additionally, we plan to investigate how the experimentation process model can be modified to seamlessly include both MABs and A/B experiments, thus preventing pitfalls from the incorrect utilization of one algorithm in an incorrect process model.

8 A trustworthy experimentation process

This chapter is based on the following publication:

An Activity and Metric Model for Online Controlled Experiments
David Issa Mattos, Pavel Dmitriev, Aleksander Fabijan, Jan. Bosch, and Helena Holmström Olsson
to appear in the Proceedings of the 19th International Conference on Product-Focused Software Process Improvement, 2018.

Chapter summary
Accurate prioritization of efforts in product and services development is critical to the success of every company. Online controlled experiments, also known as A/B tests, enable software companies to establish causal relationships between changes in their systems and the movements in the metrics. By experimenting, product development can be directed towards identifying and delivering value. Previous research stresses the need for data-driven development and experimentation. However, the level of granularity in which existing models explain the experimentation process is neither sufficient nor scalable in an online setting. Based on a case study of multiple products running online controlled experiments at Microsoft, we provide an experimentation framework composed of two detailed experimentation models focused on two main aspects; the experimentation activities and the experimentation metrics. This work intends to provide guidelines to companies and practitioners on how to set and organize experimentation activities for running trustworthy online controlled experiments.

8.1 Introduction

Prioritizing the development of software features and services that deliver value to customers is critical for the success of every company. One way to accurately discover what customers value is to evaluate the company's assumptions by means of experiments. These experiments, commonly called A/B tests, provide a framework for companies to establish causal relationships between modifications on their systems and changes in metrics. Running experiments allows companies to continuously update their assumptions on their user behavior and preferences, along with many other benefits [24].

Several publications and reports from companies such as Microsoft, Facebook, Google, and LinkedIn, among many others [17], [21]–[23], report the competitive advantage that online controlled experiments, such as A/B testing, deliver [24]. Data-driven organizations make use of relevant collected data to drive decisions and directions for the organization, and experiments are one of the key techniques used by these organizations. However, the support and evolution of experimentation practices is far from simple, and several pitfalls can invalidate experiments and lead to incorrect conclusions [64].

Different models proposed in the literature [3], [45], [46] provide a general structure for data-driven development and experiment processes. Although these models can be used as a starting point for companies to move to an iterative experiment-driven development process, previous research [6], [25], [39], [64], [65], [143], [144] also describes pitfalls, techniques to provide scaling of the experimentation process and techniques to ensure trustworthiness in the experimentation process that are not captured in or represented by the higher level of abstraction provided in these models. Because these models do not capture this level of detail, instantiating these models directly from a higher level of abstraction can lead to the limitations in the

scalability and trustworthiness of the experimentation's activities already identified by previous research. Additionally, it can lead to multiple experimentation initiatives inside an organization and lack of rigor in the process, resulting in non-comparable tests and untrustworthy results.

To address this gap, this research provides a framework that captures specific experimentation details and necessary steps for running trustworthy online controlled experiments. The framework divides the experimentation process into two main interconnected models: the set of activities that organizations should support to run trustworthy experiments, and the role of metrics and how they align experiments with long-term business goals. The proposed framework is based on an inductive case study in collaboration with the Analysis and Experimentation team at Microsoft.

The contribution of this work is twofold. First, we present the new findings from the case study. These findings represent important characteristics of the experimentation process that were not captured in previous models and reinforces the need of a new experimentation process model. Second, we present a framework composed of two models for an experimentation process that covers the two main aspects: (1) the experimentation activities and (2) the experimentation metrics. The framework provides a detailed process which aims to help companies scale their experimentation organization with a trustworthy experimentation process.

The rest of this chapter is organized as follows. Section 8.2 describes the research process of this case study. Section 8.3 presents new findings from the case study that reinforce the need for a new experimentation process model. Section 8.4 presents the two main aspects of the developed experimentation process framework. Section 8.5 concludes this chapter.

8.2 Research Method

To help companies develop and support their experimentation process and infrastructure models we conducted an inductive case study [71] in collaboration with the Analysis and Experimentation team at Microsoft.

8.2.1 Data collection

The collected empirical data consists of interview notes, whiteboard drawings, quotes and shared supporting data from nine semi-structured interviews with an average and median length of thirty-two minutes each. At the time of the data collection, the second author was working with the Analysis and Experimentation team and was the main contact person for the other researchers during the data collection and analysis phases. All the interviews were conducted in the premises of the company by the first author, who was accompanied by the second author when possible. The interviewees were selected by the second author and represent a diverse selection of software engineers and data scientists working both within the experimentation platform and as users of the platform in different product groups.

The interviews were based on a questionnaire containing eight open-ended questions, starting with a short introduction and explanation of the research. The participants were asked to describe their experimentation process. Next, the participants were asked to compare their own experimentation process and infrastructure with the existing models in the literature. Next, they were asked about what experimentation activities they performed, the time spent on these activities, their relative impact on the experiment reliability, and the other main activities performed by other people in an experiment lifecycle (from hypothesis generation to decision). We organized the interviews by products and by an approximation of the number of experiments the interviewees ran each year. This allowed us to differentiate experiences from products that run experiments on a large scale from products that start and run experiments on a small scale.

8.2.2 Data Analysis

Thematic coding was used to analyze the grouped data [73]. Recurring codes, drawings of the experimentation process and references to different parts of the platform architecture and activities helped to formalize the new findings and derive the proposed experimentation framework. For example: descriptions of different tests and analysis to ensure the experiment was properly configured were grouped in the "Pre-quality checks" while the different techniques used to evolve a metric were divided between "Online evaluation" and "Offline validation"

8.2.3 Validity considerations

To improve the construct validity of the study, and prior to the data collection stage, the semi-structured interview guide was applied to a group of two developers from a Brazilian company with experience in A/B testing, known to the first author, and two Ph.D. students in software engineering. This helped to identify potential problems, such as ambiguity in the questions and the explanations. Regarding the external validity process, although our work was conducted with only one case company, our empirical data was collected from experiences of several different products which are running trustworthy experiments at scale as well as only a few experiments per year. This helped to identify and compare trustworthy experimentation processes at different stages of maturity. Therefore, we believe that our results can be generalized for companies that want to scale their experimentation organization with a trustworthy experimentation process.

8.3 New Findings

In this section, we describe some new findings obtained from the empirical data that are not presented in previous research. Together with the description of the experimentation process collected during the interviews, these findings reinforced our motivation to develop the experimentation process framework presented in the next section.

8.3.1 Customer feedback is an important source of experimentation ideas

8.3.1.1 Description

The first finding from the empirical data is that experimentation ideas, which are later synthesized into experimentation hypotheses, are often inspired by customer feedback, instead of high-level business goals. In this research, we differentiate experiment ideas from experiment hypotheses. Experiment ideas are the first source of potential changes that can be made to systems, and they represent the potential value of a modification. However, often experiment ideas do not represent real value and therefore need to be tested in experiments [145]. Experiment hypotheses synthesize ideas into concrete experimentation scenarios, addressing what the change is, and how it can be implemented and evaluated. Ideas are synthesized into an experiment hypothesis by experiment owners. An experiment hypothesis, after deployment, can measure the real value of the synthesized idea. Developers and product owners often collect experiment ideas using different qualitative feedback collection techniques. Experiment ideas are refined, developed, prioritized and synthesized based on the experiment owners (developers, product owners and data scientists) being convinced of the positive impact for the user and for the key metrics. Another source of customer feedback ideas are differentiator features and user feedback in competitors' open channels. Although experiment ideas can come from business strategies, often they influence the prioritization process of experiments by influencing the metrics.

"It is very rare that an experiment comes from the business. Most experiments come from a group of engaged developers willing to code new ideas and run the experiments […] The ideas

for the new features and their experiments are almost always inspired by customers and competition" – Principal Data Scientist

8.3.1.2 How it differs from the existing models

Both the HYPEX and the RIGHT model propose that hypotheses are identified and prioritized based on business strategy and business goals. Although business strategy influences the hypotheses' identification and prioritization, its influence is mostly indirect, occurring via experiment metrics. The QCD model proposes business strategies, innovation initiatives, customer feedback and previous experiments as the source of new hypotheses and experiment ideas. However, it does not consider differences between experiment ideas and experiment hypotheses and how they are prioritized.

8.3.2 Metrics guide experiments towards long-term goals and help prioritize hypotheses

8.3.2.1 Description

The second finding refers to the role of metrics in learning and in hypothesis prioritization. Experiments are launched with the goal of validating a change in the system or learning more about user behavior. Both the validation of a change and the outcomes of an experiment are closely related to the validity of the metric, whether it measures its concept correctly, and whether it reflects the business strategy of the company. If a metric is misaligned with the business strategy of the company, changes and knowledge gained from experiments will also be misaligned with the strategy. However, correctly chosen metrics incentivize teams to take actions which are aligned with the company's long-term goals [114]. Good metrics will help them prioritize experiments that can have a positive impact on the long-term goals.

"If our decisions to ship are based on these metrics, these metrics have a big impact on the development and evolution of the product. They guide the teams to develop and focus their work to improve these metrics" – Principal Data Scientist

8.3.2.2 How it differs from the existing models

Existing experimentation models do not describe how metrics impact hypothesis prioritization and long-term company goals. Metrics are considered as part of the instrumentation system, which does not reflect the bidirectional influence on the business strategy of the company.

8.3.3 Metrics evolve and capture the experiment assumptions

8.3.3.1 Description

As discussed in the second finding, metrics can guide the company's long term-goals and help prioritize the hypotheses. Additionally, metrics also capture uncertainty and experiment assumptions. As metrics often represent abstract and subjective concepts [114] such as satisfaction and engagement, they contain assumptions about what constitutes these concepts. These assumptions should be tested and validated during the experimentation process and should be constantly iterated in order to maintain alignment with the business strategy. This constant update and validation of the assumptions based on the results of an experiment leads to a metric evolution process. Metrics can start as low level signals, and then evolve to capture more complex concepts that are more closely aligned with the business strategy. Additionally, the evolution of metrics also reflects the evolution of the business strategy and the product focus over time. As the company changes its strategies, metrics should be updated to align with these changes.

8.3.3.2 How it differs from the existing model

Existing models describe the presence of uncertainty and assumptions in the business strategy and in the role of the business analyst, as they are responsible for hypothesis prioritization. However, these models do not capture how the metrics evolution and the business strategy influence each other and impact the product.

8.4 The experimentation process framework

In the previous section, we discussed the findings and compared them to previous research. Previous research did not capture all the identified characteristics from the findings nor present specific experimentation details, or the necessary steps to take when running trustworthy online controlled experiments. This led to the motivation and the need to develop a new framework that incorporates these findings. The framework is based on the collected empirical data, including descriptions and drawings of the process, comparison with other processes and the characteristics identified in the new findings and in the different components represented in the Microsoft ExP Platform described in [146]. During the data collection, the researchers asked the interviewees to describe their experimentation process and compare it with existing models. This discussion, together with findings from previous research, led to the development of this framework, which consists of the two main aspects of the experimentation process: (1) the experimentation activities, and (2) the experimentation metrics. These aspects are related to the business long-term strategies as represented in Figure 8.1. This diagram represents how the two main aspects of experimentation used in this work relate to each other and to the business, and how they connect to the findings discussed in the previous section. Next, we detail these two aspects in two separate models, the experimentation activity model, and the metric model. These models were developed based on the collected empirical data and previous research.

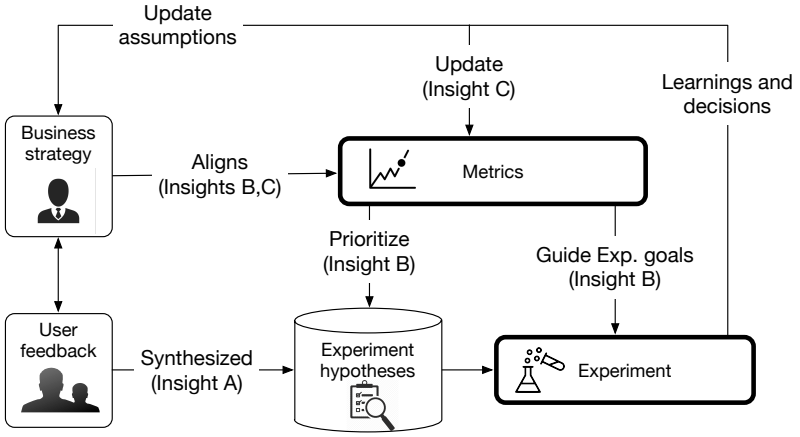

Figure 8.1 - The relationship between the two main aspects (in bold), their relation with the business strategy, and how the findings connect them.

8.4.1 The experimentation activity model

The experimentation activity model describes the different activities which comprise a single experiment iteration, from the experiment ideas, to the experiment analysis necessary for the running of trustworthy online experiments. The arrows correspond to sequential connected activities. For example, when the experiment is launched pre-quality checks are run followed by ramp-up procedures before the experiment data is collected from a larger user base. Furthermore, our model divides the experimentation process into three sequential phases: *the experiment development phase*, *the experiment execution phase,* and *the experiment analysis phase*. We illustrate the experimentation activity model in Figure 8.2 and describe the three phases in greater detail.

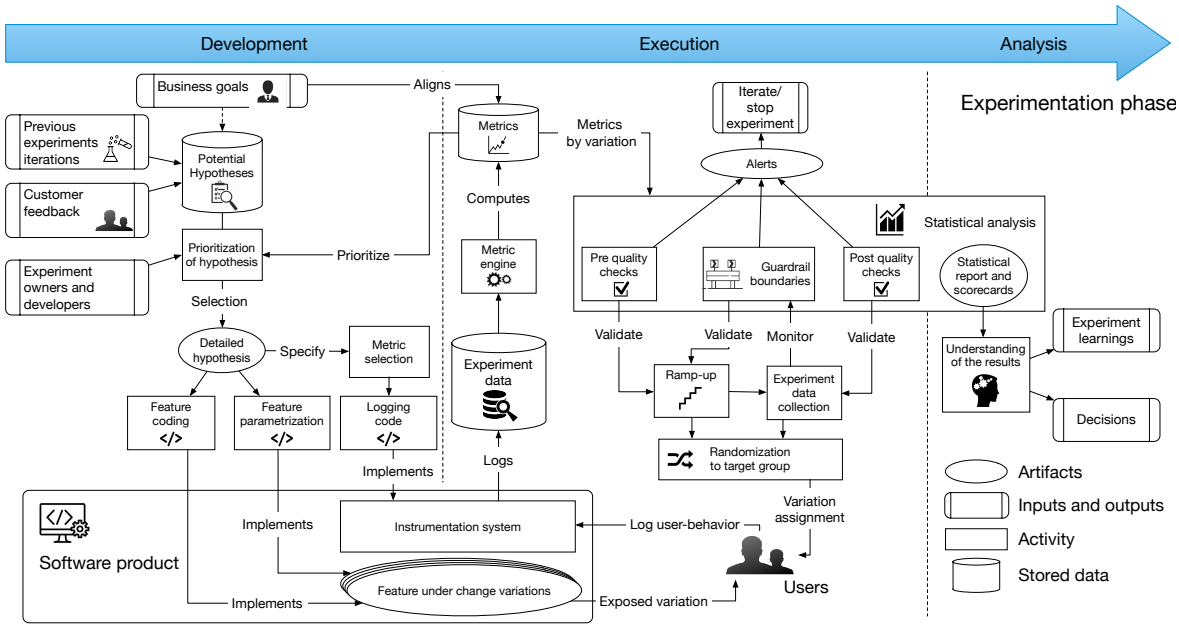

Figure 8.2 - The experimentation activity model

8.4.1.1 Experiment development phase

This phase refers to the specification and development activities of the experiment necessary to implement the variation change. This phase takes place before users are exposed to any variations. Finding A discusses how experiment hypotheses are generated and synthesized in the experimentation process. Experiment ideas are usually derived from four sources: (1) customer feedback (Finding A), (2) further iteration from previous experiments iterations [38], (3) need to understand and model the user behavior, and (4) less often through higher-level business goals (dashed line). The hypotheses are prioritized based on the experiment owners' prior analysis of how the hypotheses can impact the OEC. This analysis can be based on historical data from the feature, insights from other market segments (e.g. a feature that shows success in the US market might be prioritized for launch in another market or globally), or on experience gained from similar previous experiments or other products.

Following prioritization of the hypothesis a detailed hypothesis is elaborated. This includes specification of the experiment type (A/B, A/B/n, MVT etc.), how many variants are going to be present, cohorts or market segmentation, experiment duration and metrics to be used. Additionally, it covers the feature/change specification, including the area of functionality, actual and expected behavior, and implementation alternatives and considerations.

In addition, the detailed hypothesis specifies the target metrics that are expected to have impact and movements. The specification of the experiment metrics is closely related to the metrics used to prioritize the experiment itself. This includes lower-level signals that measure user-specific behaviors, guardrail metrics that indicate whether an experiment is within the allowed experiment conditions, and the leading metrics [24] that guide the experiment analysis. The metric selection is related to logging capabilities. The logging code represents the instrumentation of the system that interacts with the experimentation system. The logging code collects lower-level user behavior signals that can be used to compose complex metrics in the experimentation system. It is worth noting that the logging code should comply with the same standards (e.g. deployment, testing, code review, and integration pipeline) as any other product code.

Depending on the type of detailed specification of the hypothesis, the change in the system at the product level can happen in two ways, or a combination of the two. The first is through coding of the modification. This method is common when the experiment specification requires coding of a new feature or extensive modification of existing ones. The experiment set-up is a comparison of the current system with the change (treatment) and the system without the change (control). The second way in which the change can be done is by parametrizing an existing functionality and running experiments to modify these parameters. In this case the functionality already exists but appears to have suboptimal performance. The parameters of this functionality are configured during the experiment execution and are assigned to users. Although this might require a larger overhead in supporting and setting up a configuration manager, it reduces the effort and time spent on each experiment.

8.4.1.2 Experiment execution phase

After the experiment is properly designed, the metrics selected, the change coded and instrumented, the experiment moves to the experiment execution phase. Here, users are randomized and are assigned to the experiment variations of the feature under change. The user behavior is logged, and the initial metrics are computed per variation using the metric engine and initial statistical analysis are computed using the statistical analysis tool. The metric engine is responsible for collecting and transforming raw data into experiment metrics. These metrics are consumed by the statistical analysis tool in order to run quality checks, check for guardrail-metrics and generate scorecards. The metrics, as discussed in the metric model, align the experiment goals with the business strategy and serve as input in the prioritization of experiment hypotheses.

The assignment of the users to the different variations can happened in two general ways (other methods used for websites are described in [6]). The first is to use a feature toggling system. In this case, the change in the system is parametrized using a variable, and the users are assigned to feature variations that the parameter activates (treatment group) or deactivates (control group). The second method is through traffic routing, where multiple instances of the system are run in parallel and the assignment system redirects the user to one of the multiple instances. The randomization refers to how users are assigned to a specific variation of the experiment during the execution phase. The randomization usually targets a specific group of users at the beginning and then generalizes to a larger audience. This target is specified in the detailed hypothesis. Although randomization might look intuitive, there are several techniques available to ensure that the randomization is not biased towards any variation [6], [67]. Then the instrumentation system captures the user behavior and logs the experiment data for use in statistical analysis.

The first step in the execution phase is to have confidence that the experiment will yield trustworthy results.

"A lot of effort goes into making sure the experiment passes the (pre-) quality checks. This is an essential step that gives us confidence in the experiment, so that we will not go to the next steps only to discover we did something wrong at the beginning" – Principal Data Scientist

Before running experiments and exposing users to different variations, pre-quality tests are run to check for common pitfalls. Examples of pre-quality tests are: A/A or null tests, sample ratio mismatching, randomization checks, and offline testing. The A/A test assigns the users to the same variant A (the system without the change) with the aim of testing the experimentation system and assessing variability in the collected data [6]. The sample ratio mismatch (SRM) [39] is considered a critical diagnosis tool for online experiments. The SRM checks the percentage allocation of the users. This allows the experiment owners to detect bias (that would invalidate the experiment results) towards any variation as well as check performance

considerations. Randomization checks are tests which identify whether the randomization procedure is biased or has any patterns and checks the consistency of the randomization between sessions (to ensure that recurring users see the same variations). Offline testing uses historical data to assess the impact of the changes in the system and estimate confidence intervals [34].

After the pre-quality checks the activities that take place in the experiment execution phase are: the ramp-up, guardrail boundaries and experiment data collection. Ramp-up is a procedure where the treatment variations are initially launched to a small percentage of users. This is useful because critical errors can be detected early while exposing only a small number of users to the treatment variations. Large effect sizes in key metrics are mostly related to experiment errors [108], therefore fewer users' needs will be exposed to the change while identifying such errors. As the experiment runs without severe degradations, the percentage of users exposed to the treatment can be continuously increased until each variation has equal allocation, so that the experiment power is maximized [6]. A ramp-up procedure should be implemented together with an automated alerting capability with different significant levels and configurable actions. By checking guardrail metrics and experiment boundaries, such as key metrics that the experiment should not alter or deteriorate, the alerting system will alert experiment owners if something unusual is happening. In extreme cases, it will shut down an experiment with a significant negative impact if no action was taken. This allows organizations to invest in innovative and bold changes while reducing the risk of exposing users to bad ideas and errors [6].

After the main experiment execution, post-quality tests can be run to ensure that the experiment is valid and the data is reliable. Common post-quality checks are (1) checking for experiment invariant metrics, (2) learning effects, (3) A/A tests, (4) interaction effects with other overlapping experiments, and (5) novelty effects [26]. Experiment invariant metrics are metrics that are not expected to change within the scope of the experiment. If there is a statistically significant movement in those metrics during the experiment execution, either the assumptions about the impact of the experiment are wrong or the implementation of the experiment is wrong. In both cases, it is worth exploring the reasons for these unexpected results. Other quality checks that are beyond the scope of this work can be seen in [17], [26], [39]. The statistical analysis tool supports the whole experiment execution, computing guardrail tests and quality checks. Following the post-quality checks, the statistical analysis tool generates reports and scorecards for the key metrics of the experiment. Qualitative data collected from feedback boxes and other consumer feedback channels (if available) can be used together with the quantitative analysis to help explain the result.

8.4.1.3 Experiment analysis phase

The analysis phase follows both the data collection and the conclusion of the experiment execution. The analysis phase consists of developing an understanding of the statistical output of the experimentation system in the context of assumptions about user behavior.

Understanding of the results is an activity that analyzes the results from the statistical analysis in order to generate evidence about customer preferences and behavior and thus facilitate the decision-making process. It is important to reinforce that as the complexity of the experiment increases and there is not a standard OEC, key metrics can move in opposite directions, behave differently in different markets and in different user segments. In such scenarios, it is important to understand why different markets or user segments behaved differently. Not only does this generate meaningful knowledge which can be used to update assumptions about user behavior, but it also facilitates the process of decision-making and helps the evolution of the metrics and their alignment with the business strategy. Based on the results of this activity, the company

can make decisions (such as ship or not ship the change) and update their assumptions and the metrics.

8.4.2 The experiment metric model

A key component of the experimentation process described above is the metrics. Metrics guide hypothesis prioritization, the instrumentation required in the system, and the understanding of the results, and reveal whether the experiment results can be trusted. The experiment metric model that we discuss in this section characterizes two aspects of metrics: metric lifecycle and metric type. These two aspects are related to the findings B and C. These findings reinforce the central role that metrics play in the experiment design and execution. The metric lifecycle is divided into four main phases: creation, evolution, maturity and phase-out. The metric type is based on the four metric types identified in previous research [39]: OEC metrics, data-quality metrics, guardrail metrics and local feature and diagnostic metrics. The metric model is represented in Figure 8.3.

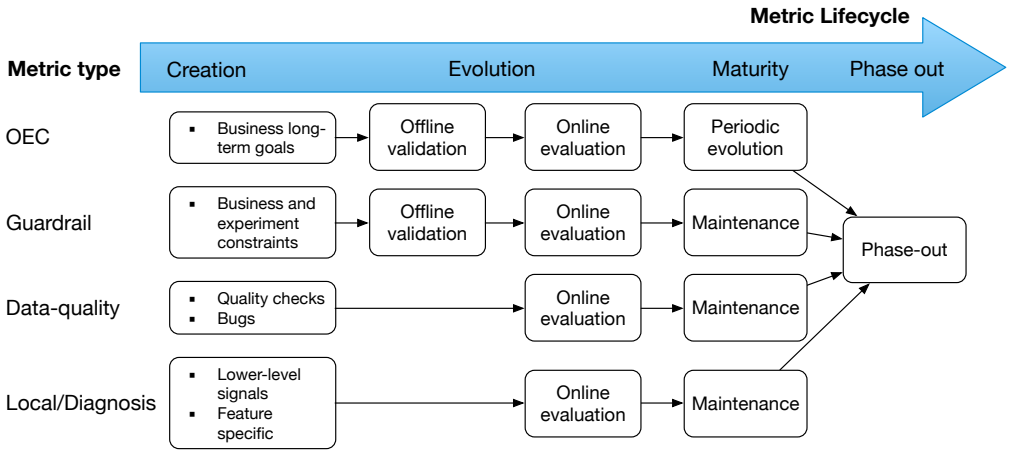

Figure 8.3 - The metric model

The arrow in the metric model refers to the different stages in a metric lifecycle. In the creation phase, a first prototype of a metric is created. In this step, the metric consists of either aggregated lower-level signals (such as usage time, clicks, etc.), or modifications of existing metrics (such as linear combination or proportions).

The evolution phase consists of refining the metric and aligning it with the metric goal. During this process additional metrics can be combined with the original one to better capture more complex concepts. The refinement process can also impact the sensitivity of the metric. Offline validation and online evaluation are two techniques used to assess and support the evolution of a metric. The offline validation process analyses the metric directionality and sensitivity. Directionality refers to the direction of positive impact. The sensitivity of a metric refers to how well a metric is capable of moving due to the treatment. Techniques for offline validation of metrics can be found in [40], [114]. Online evaluation refers to analyzing the metric during an experimental run. This includes computing the metric with live users. The evaluation can be done through the comparison of the new metric with other existing metrics and through degradation experiments. Degradation experiments refer to the degrading of the user experience to find the directionality and sensitivity of metrics in the absence of an experiment corpus or analogous metrics for comparison. Strange movements of metrics during the experiment execution and quality tests can indicate instrumentation problems.

The maturity phase represents a period where the metric has been evaluated or validated and does not go through extensive modifications. For some metrics, the maturity phase represents a phase where they are updated in pre-established periods with learnings from multiple experiments or updated to accommodate changes in the business strategy of the company or the product, or only when issues arise in a maintenance process.

The last phase is the phase-out. In this phase, older metrics can be replaced with newer metrics, and metrics specific to an experiment or feature are deactivated after the experiment or feature lifecycle is over. It is worth noting that each metric might reach the different stages during different time frames. Metrics designed to be used only in one experiment can go through the creation to the phase-out process in only one experiment cycle. Metrics specific to features go through many experiment cycles, until the feature is only maintained or it is abandoned. OEC metrics that cross several features and even products can last many experimental cycles and years.

The second aspect of the metric model that we describe refers to the type of metric. Overall Evaluation Criteria (OEC) metrics guide the experiment outcomes and are a measure of the experiment's success. They represent and capture assumptions about business strategies and long-term company goals. OEC metrics are used across experiments and their evolution depends on the inputs from multiple experiments and the alignment with business goals. The evolution of OEC metrics affects multiple experiments and therefore is only updated periodically. The update of such metrics goes through offline validation and online evaluation.

Data-quality metrics are used in quality checks and inconsistency checks, such as implementation bugs, synchronization errors, and telemetry loss. Some of these metrics are feature specific, such as checking for data quality in experiments specific for a feature, while others are used in multiple experiments during pre-quality checks, such as the Sample Ratio Mismatch and checks for randomization imbalance during A/A tests. These metrics are evaluated online and their evolution and update occur when feature-specific modifications require updates or when issues arise.

Guardrail metrics are metrics that are not used as an indicator of success but instead serve as boundaries for the experiment. Negative movements of guardrail metrics might be an indicator that experiment conditions were violated, generating alerts. These metrics, although they do not represent business directions as the OECs do, can represent business constraints on the OEC movement. These metrics evolve periodically in order to align with changing business restrictions. When updated, guardrail metrics go through offline validation and online evaluation.

Local feature and diagnosis metrics are metrics used in individual functionalities of products. They do not impact other experiments and serve as diagnostic indicators used to understand the movement of OECs and guardrail metrics. Diagnosis metrics represent lower-level signals and serve as debug metrics for understanding unexpected movements of OEC and guardrail metrics. Local feature and diagnosis metrics are usually constrained to the experiment or feature lifecycle. Due to their short lifecycle these metrics are only evaluated online. To support the creation, evolution, maturity and phase-out phases of the different types of metrics, the experimentation team should support a metric management platform. This type of system prioritizes important metrics, constrains metrics to specific features and keeps track of inactive phased-out metrics for comparison and offline validation between older and newer experiments.

8.5 Conclusion

Online controlled experiments have become the standard practice for evaluating ideas and prioritizing features in most large web-facing software-intensive companies [21]–[24].

Although companies can rely on models to start their experimentation organization and data-driven practices they might struggle to establish a trustworthy experimentation process as they scale their experimentation organization. Previous research provides models and processes for starting an experimentation organization based on higher-level descriptions of the experimentation process, but they done capture all the pitfalls and techniques that allow the companies to scale and to ensure trustworthiness in the experimentation process [6], [25], [39], [64], [65], [143], [144]. Based on case study research with multiple product teams responsible for running online controlled experiments at Microsoft, we provide an experimentation framework composed of two detailed experimentation models focused on two main aspects; the experimentation activities and the experimentation metrics. This model discusses granular aspects of the experimentation process that can help companies and practitioners to scale their experimentation activities into a trustworthy experimentation process.

In future research, we plan to validate this experimentation process in other companies, to compare how the different activities map onto their experimentation process and provide further refinement. We also plan to conduct further studies into other aspects of the experimentation process, such as how the organization roles change during the evolution of the experiment and how experiment artifacts are managed and evolve during the experiment organization.

9 Conclusion

This thesis explores the topic of automated online experiments from the perspectives of software architecture, algorithms for experimentation and the experimentation process in the scope of the presented goals and research questions. This chapter is organized in three sections. First, it provides an overview of the research question and how each of them is discussed and answered in Chapters 4 to 8. Second, it provides an overview of the key contributions of this thesis in relation the proposed research goals. Third, it concludes with a discussion of future work.

9.1 Overview of the research questions

In this section, we present the overview of the research question and how each of them is discussed and answered in Chapters 4 to 8.

RQ1: How to architect an automated experimentation system?

RQ1.a: What are architecture qualities that can support an automated experimentation system?

RQ1.b: What are the different architectural design decisions for an automated experimentation system?

The RQ1 is discussed in Chapter 4. We developed an architecture framework for an experimentation system, discussing architectural qualities and design decisions. It was presented the architectural qualities for an automated experimentation system (the external adaptation, the data collection as integral part of the architecture, performance reflection, explicit learning system, decentralized adaptation and exchange of knowledge between the different involved systems). In respect to RQ2.b, we present a justification of the architectural design decisions together with a discussion of alternative implementations regarding to the different presented problems: the usage of guardrail metrics and information exchange; dealing with confounding factors through centralization coordination mechanisms, architecture centralization and type of experimentation integration.

The developed architecture framework serves as a reference for other automated experimentation systems and it was instantiated in a few different cases. The chapter presents instantiation in a mobile autonomous robot, to solve a particular human-robot proxemics distance problem. This architecture was implemented in Chapter 6 in collaboration with Sony Mobile in a mobile application domain, and it is being used together with embedded systems companies as an optimization through experiments service.

This architecture is also being used in our current and future research and has expanded to be able to (1) integrate in multiple software platforms by providing a generic interface, (2) to run multiple types of algorithms, including A/B/n, full-factorial, multi-armed bandits, continuous-armed bandits and contextual bandits and (3) to be used by developers not familiar with these algorithms. Gerostathopoulos et al. [147], [148] provide another custom instantiation of this architecture framework. In this work, the architecture is used for optimization of warmup behavior with JIT compilation and for optimization of city traffic navigation.

RQ2: How can embedded systems industry adopt continuous experimentation in their development process?

RQ2.a: What are the recognized challenges towards continuous experimentation faced by the embedded systems industry?

RQ2.b: What are the recommended strategies to facilitate the use of continuous experimentation in the embedded systems domain?

The RQ2 is addressed in Chapter 5 through different challenges that embedded systems face when adopting continuous experimentation. For RQ2.a, this chapter provides a list of twelve challenges classified in terms of technical, business and organizational challenges. These challenges are: expensive testing scenarios, real-time and safety constraints, lack of over-the-air updates, lack of experimentation tools for the domain, challenge in managing multiple stakeholders in the experimentation process, repetitive tuning performed by experts, top-down decisions, long release cycles, lack of sharing B2B data, metrics validation, lack of data insights and privacy assurance. In RQ2.b we answer how each of these challenges can be addressed with a set of strategies. The strategies are grouped in three main groups: architectural changes, data handling and changes in the development process. The relation between the strategies and the challenges can be seen in Figure 5.1.

One can infer that the strategies indicate the need for novel techniques and automated experiments might be a key technique for faster adoption of experiments in the embedded systems domain. One of the direct outcomes of these results was a broader understanding of the requirements to develop an experimentation system that could be used by embedded systems. This lead to a different instantiation of the architecture framework discussed in Chapter 4, that is now being used in our current and future research as discussed earlier.

RQ3: How can an automated experimentation system be integrated with existing systems to optimize feature parameters?

RQ3 is partially answered in Chapter 5, where we instantiate the architecture framework, discussed in Chapter 4, in collaboration with Sony Mobile. Some of the architectural qualities such as external adaptation, exchange of knowledge and explicit learning component allowed easy integration and verification of the experimentation system with the existing application.

Existing systems also aim at improving features that are not in a discreate space. Optimization in the continuous space poses a limitation for these systems. Based on that, we propose a modification of the HOO algorithm for continuous-armed bandits creating the LG-HOO algorithm. This new algorithm is shown to have better time and decision properties compared to the original algorithm and help us to tackle the discrete space limitation of current optimization techniques

RQ4: What are the restrictions and pitfalls associated with multi-armed bandit algorithms in online experiments and how to overcome them?

The RQ4 is addressed in Chapter 7. We present pitfalls and strategies often seen in the application of multi-armed bandit algorithms to online systems as well as strategies used by the case study companies. The identified pitfalls are: increase in the type I error in MAB experiments, due to naïve implementations, violation of assumptions and usage of MAB in exploration problems. The second problem consists of detecting experimentation problems due to the lack of quality checks in MAB, presence of Simpson's Paradox and increased complexity in ramp-up procedures. The third problem is the increased complexity of the experiment design due to the adaptive allocation of the metrics, user consistency problems and communication of the experiment results. The strategies to avoid these pitfalls often gravitated in either falling back to traditional experiment methods, mixing traditional design methods to overcome some of the restrictions or addressing the problem through a statistical modeling and refinement of the MAB algorithm.

RQ5: What are the necessary steps and the set of activities in a trustworthy experimentation process?

The RQ5 is addressed in Chapter 8. This chapter analyzes a trustworthy state-of-the-art experimentation process to understand the steps and activities that they involve. We propose an experimentation process framework composed of two models, an activity model and a metric model. These two models provide a greater level of detail and description of the experimentation process compared to previous research. Although this chapter does not deal directly with the automated experimentation topic, it serves as basis for our current and future work, to understand what parts of the process impact the experimentation cost and would benefit from automation; and how these different parts could be integrated in a unified automated experimentation system that has the same or higher level of the current state-of-the-art experimentation practices.

9.2 Key contributions

The key contributions of this thesis to the development and understanding of automated experiments in relation to the research goals are:

Goal 1 – To analyze how automated experiments can be supported from a software architecture perspective

- Identification of software qualities to support automated experimentation.
- Description of a set of architectural design decisions for an automated experimentation system.
- Proposal of an initial architecture framework for automated experimentation that was evaluated in an autonomous mobile robot.
- Identification of key research challenges that need to be addressed to further support the development of automated experimentation.

Goal 2 – To identify how existing systems, including embedded systems, can use continuous experimentation to optimize software features

- Identification of the key challenges faced by embedded systems companies when adopting continuous experimentation and proposal of a set of strategies and guidelines to overcome these challenges.

Goal 3 – To analyze how different algorithms for automating experiments influence the outcome of an experiment

- Development of a bandit algorithm for continuous space parameters (LG-HOO) that considers online experimentation constrains and statistical evidence that supports the usage of the algorithm in online experimentation scenarios.
- Validation of the LG-HOO algorithm in an industrial case study.
- Identification and analysis of restrictions and pitfalls of multi-armed bandit algorithms applied to online experiments.
- Identification of strategies for practitioners to avoid pitfalls in online controlled experiments.
- A set of guidelines for practitioners to select an experimentation technique that minimizes the occurrence of the identified pitfalls.

Goal 4 – To identify what are the different parts of the experimentation process, the costs and how to automate the steps in the experimentation process

- Identification of the differences between existing experimentation models and the current state-of-the art.
- The proposal of a new experimentation process framework consisting of two models, an activity model and an experiment metric model. This framework captures and explain the experimentation process in a level of detail not provided by any of the experimentation models discussed in previous research.

9.3 Future work

In our research, we explored different aspects of online experiments with the aim of creating a better understanding on how to automate different types of experiments and how companies can support and optimize their systems through automated experiments.

Our ongoing research studies the implementation and application of the framework described in Chapter 4 in embedded systems industry. Our goal is to understand how this domain can incorporate an optimization through experiment systems in their development process without compromising existing real-time and safety requirements. Although experimentation is traditionally used with customer behavior and preferences, we believe a similar approach can be used to optimize the performance of stochastic systems in offline software and hardware-in-the-loop scenarios. Similar approach has been proposed in [27] and evaluated in [147]. We plan to investigate if successful implementations of this approach in the embedded systems industry can trigger the usage experiment-based development techniques.

Currently, automated experiments are fairly limited in scope. As identified in Chapter 8, other parts beyond the execution impact in the cost of each experiment iteration. Future work needs to better understand the impact and costs of each experiment metric and experiment activity and analyze the return on investment of automation of these parts. This analysis can lead to more focused efforts in automating experiments and development of experimentation systems.

Bibliography

[1] P. W. Devine, C. A. Srinivasan, and M. S. Zaman, "Importance of Data in Decision-Making," in *Business Intelligence Techniques*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 21–39.

[2] A. Fabijan, H. H. Olsson, and J. Bosch, "Customer Feedback and Data Collection Techniques in Software R&D: A Literature Review," in *Software Business, ICSOB 2015*, vol. 210, J. M. Fernandes, R. J. Machado, and K. Wnuk, Eds. Braga, Portugal: Springer International Publishing, 2015, pp. 139–153.

[3] F. Fagerholm, A. Sanchez Guinea, H. Mäenpää, and J. Münch, "The RIGHT model for Continuous Experimentation," *J. Syst. Softw.*, vol. 123, pp. 292–305, 2017.

[4] F. Fagerholm, A. S. Guinea, H. Mäenpää, and J. Münch, "Building Blocks for Continuous Experimentation," *Proc. 1st Int. Work. Rapid Contin. Softw. Eng.*, pp. 26–35, 2014.

[5] A. Fabijan, H. H. Olsson, and J. Bosch, "Time to Say 'Good Bye': Feature Lifecycle," in *2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2016, pp. 9–16.

[6] R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne, "Controlled experiments on the web: survey and practical guide," *Data Min. Knowl. Discov.*, vol. 18, no. 1, pp. 140–181, Feb. 2009.

[7] M. E. Porter, *Competitive Advantage*. 1980.

[8] A. F. Payne, K. Storbacka, and P. Frow, "Managing the co-creation of value," *J. Acad. Mark. Sci.*, vol. 36, no. 1, pp. 83–96, 2008.

[9] J. Bosch, "Building products as innovation experiment systems," *Lect. Notes Bus. Inf. Process.*, vol. 114 LNBIP, pp. 27–39, 2012.

[10] P. Bosch-Sijtsema and J. Bosch, "User Involvement throughout the Innovation Process in High-Tech Industries," *J. Prod. Innov. Manag.*, vol. 32, no. 5, pp. 793–807, 2015.

[11] A. Fabijan, H. H. Olsson, and J. Bosch, "Early value argumentation and prediction: An iterative approach to quantifying feature value," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9459, no. JANUARY, pp. 16–23, 2015.

[12] A. Fabijan, H. H. Olsson, and J. Bosch, "Customer Feedback and Data Collection Techniques in Software R&D: A Literature Review," in *Icsob*, vol. 114, 2015, pp. 139–153.

[13] B. Meyer, *Agile!: The good, the hype and the ugly*, 1st ed., vol. 9783319051. Cham: Springer International Publishing, 2014.

[14] E. Ries, *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. 2011.

[15] M. Fowler, "Continuous Integration," *Integr. Vlsi J.*, vol. 26, no. 1, pp. 1–6, 2006.

[16] D. G. Feitelson, E. Frachtenberg, and K. L. Beck, "Development and deployment at facebook," *IEEE Internet Comput.*, vol. 17, no. 4, pp. 8–17, 2013.

[17] R. Kohavi, A. Deng, B. Frasca, R. Longbotham, T. Walker, and Y. Xu, "Trustworthy online controlled experiments," *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discov. data Min. - KDD '12*, p. 786, 2012.

[18] H. H. Olsson and J. Bosch, "Towards Data-Driven Product Development: A Multiple Case Study on Post-deployment Data Usage in Software-Intensive Embedded Systems," *Lean Enterp. Softw. Syst. - Proc. 4th Int. Conf. Lean Enterp. Softw. Syst. LESS 2013, Galway, Ireland, December 1-4, 2013*, pp. 152–164, 2014.

[19] H. H. Olsson and J. Bosch, "Post-deployment Data Collection in Software-Intensive Embedded Products," *Contin. Softw. Eng.*, vol. 9783319112, pp. 1–226, 2014.

[20] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*, vol. 1. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[21] E. Bakshy, D. Eckles, and M. S. Bernstein, "Designing and Deploying Online Field Experiments," *Proc. 23rd Int. Conf. World wide web - WWW '14*, pp. 283–292, Sep. 2014.

[22] H. Gui, Y. Xu, A. Bhasin, and J. Han, "Network A/B Testing," in *Proceedings of the 24th International Conference on World Wide Web - WWW '15*, 2015, pp. 399–409.

[23] D. Tang, A. Agarwal, D. O'Brien, and M. Meyer, "Overlapping experiment infrastructure," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '10*, 2010, p. 17.

[24] A. Fabijan, P. Dmitriev, H. H. Olsson, and J. Bosch, "The benefits of controlled experimentation at scale," in *Proceedings - 43rd Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2017*, 2017, pp. 18–26.

[25] R. L. Kaufman, J. Pitchforth, and L. Vermeer, "Democratizing online controlled experiments at Booking.com," pp. 1–7, Oct. 2017.

[26] A. Fabijan, P. Dmitriev, H. H. Olsson, and J. Bosch, "The Evolution of Continuous Experimentation in Software Product Development," in *Proceedings of the 39th International Conference on Software Engineering ICSE'17*, 2017.

[27] J. Bosch and H. H. Olsson, "Data-driven continuous evolution of smart systems," in *Proceedings of the 11th International Workshop on Software Engineering for Adaptive and Self-Managing Systems - SEAMS '16*, 2016, pp. 28–34.

[28] Y. Xu, W. Duan, and S. Huang, "SQR: Balancing Speed, Quality and Risk in Online Experiments," no. 1, pp. 1–9, Jan. 2018.

[29] F. Yang, A. Ramdas, K. Jamieson, and M. J. Wainwright, "A framework for Multi-A(rmed)/B(andit) testing with online FDR control," no. 2, pp. 1–26, Jun. 2017.

[30] G. Schermann, D. Schöni, P. Leitner, and H. C. Gall, "Bifrost – Supporting Continuous Deployment with Automated Enactment of Multi-Phase Live Testing Strategies Gerald," *Proc. 17th Int. Middlew. Conf. - Middlew. '16*, no. December, pp. 1–14, 2016.

[31] A. Deng, J. Lu, and S. Chen, "Continuous monitoring of A/B tests without pain: Optional stopping in Bayesian testing," *Proc. - 3rd IEEE Int. Conf. Data Sci. Adv. Anal. DSAA 2016*, vol. 2, no. 3, pp. 243–252, 2016.

[32] A. Datta, M. C. Tschantz, and A. Datta, “Automated experiments on ad privacy settings: A tale of opacity, choice, and discrimination,” *Proc. Priv. Enhancing Technol.*, vol. 2015, no. 1, pp. 92–112, 2015.

[33] G. Tamburrelli and A. Margara, “Towards Automated A/B Testing,” in *International Symposium on Search Based Software Engineering*, 2014, pp. 184–198.

[34] L. Bottou, J. Peters, J. Quiñonero-Candela, D. X. Charles, D. M. Chickering, E. Portugaly, D. Ray, P. Simard, and E. Snelson, “Counterfactual Reasoning and Learning Systems,” *J. Mach. Learn. Res.*, vol. 14, pp. 3207–3260, 2013.

[35] G. Schermann, J. Cito, P. Leitner, U. Zdun, and H. C. Gall, “We’re doing it live: A multi-method empirical study on continuous experimentation,” *Inf. Softw. Technol.*, vol. 99, pp. 41–57, 2018.

[36] D. C. Montgomery, *Design and analysis of experiments*, 8th ed. John Wiley And Sons Ltd, 2012.

[37] R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne, “Controlled experiments on the web: Survey and practical guide,” *Data Min. Knowl. Discov.*, vol. 18, no. 1, pp. 140–181, 2009.

[38] K. Kevic, B. Murphy, L. Williams, and J. Beckmann, “Characterizing experimentation in continuous deployment: A case study on bing,” in *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track, ICSE-SEIP 2017*, 2017, pp. 123–132.

[39] P. Dmitriev, S. Gupta, K. Dong Woo, and G. Vaz, “A Dirty Dozen: Twelve Common Metric Interpretation Pitfalls in Online Controlled Experiments,” *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. data Min. - KDD ’17*, pp. 1427–1436, 2017.

[40] A. Deng and X. Shi, “Data-Driven Metric Development for Online Controlled Experiments,” *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. - KDD ’16*, pp. 77–86, 2016.

[41] D. I. Mattos, P. Dmitriev, A. Fabijan, J. Bosch, and H. H. Olsson, “An Activity and Metric Model for Online Controlled Experiments,” in *to appear in the Proceedings of the 19th International Conference on Product-Focused Software Process Improvement*, 2018.

[42] Eric Ries, *The Lean Startup: How Today’s Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*, 1st ed. Crown Publishing Group, 2011.

[43] J. Bosch, H. H. Olsson, J. Björk, and J. Ljungblad, “The Early Stage Software Startup Development Model: A Framework for Operationalizing Lean Principles in Software Startups,” *Lean Enterp. Softw. Syst.*, pp. 1–15, 2013.

[44] H. H. Olsson, H. Alahyari, and J. Bosch, “Climbing the ‘Stairway to heaven’ - A mulitiple-case study exploring barriers in the transition from agile development towards continuous deployment of software,” *Proc. - 38th EUROMICRO Conf. Softw. Eng. Adv. Appl. SEAA 2012*, pp. 392–399, 2012.

[45] H. H. Olsson and J. Bosch, “Towards Continuous Customer Validation: A Conceptual Model for Combining Qualitative Customer Feedback with Quantitative Customer Observation,” in *Icsob*, vol. 114, 2015, pp. 154–166.

[46] H. H. Olsson and J. Bosch, “The HYPEX Model: From Opinions to Data-Driven Software Development,” *Contin. Softw. Eng.*, vol. 9783319112, pp. 1–226, 2014.

[47] S. L. Scott, “Multi-armed bandit experiments in the online service economy,” *Appl. Stoch. Model. Bus. Ind.*, no. June 2014, pp. 1–14, 2015.

[48] W. H. Press, “Bandit solutions provide unified ethical models for randomized clinical trials and comparative effectiveness research.,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 106, no. 52, pp. 22387–92, 2009.

[49] M. Moeini, O. Wendt, and L. Krumrey, “Portfolio Optimization by Means of a $$\chi $$ -Armed Bandit Algorithm,” in *Aciids (2)*, vol. 5991, 2016, pp. 620–629.

[50] W. Shen, J. Wang, Y. G. Jiang, and H. Zha, “Portfolio choices with orthogonal bandit learning,” *IJCAI Int. Jt. Conf. Artif. Intell.*, vol. 2015–Janua, no. Ijcai, pp. 974–980, 2015.

[51] L. Li, S. Chen, J. Kleban, and A. Gupta, “Counterfactual Estimation and Optimization of Click Metrics for Search Engines,” *CoRR*, vol. abs/1403.1, pp. 929–934, 2014.

[52] L. Li, W. Chu, J. Langford, and R. E. Schapire, “A contextual-bandit approach to personalized news article recommendation,” in *Proceedings of the 19th international conference on World wide web - WWW ’10*, 2010, p. 661.

[53] G. Burtini, J. Loeppky, and R. Lawrence, “A Survey of Online Experiment Design with the Stochastic Multi-Armed Bandit,” pp. 1–49, Oct. 2015.

[54] V. Kuleshov and D. Precup, “Algorithms for the multi-armed bandit problem,” *J. Mach. Learn.*, vol. 1, pp. 1–32, 2000.

[55] J. White, *Bandit Algorithms for Website Optimization*, 1st ed. O’Reilly Media Inc, 2012.

[56] R. S. Sutton and A. G. Barto, “Sutton & Barto Book: Reinforcement Learning: An Introduction,” 1998.

[57] M. M. Drugan and A. Nowe, “Designing multi-objective multi-armed bandits algorithms: A study,” *Proc. Int. Jt. Conf. Neural Networks*, 2013.

[58] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd ed. Pearson Education, 2013.

[59] A. Jansen and J. Bosch, “Software Architecture as a Set of Architectural Design Decisions,” in *5th Working IEEE/IFIP Conference on Software Architecture (WICSA’05)*, 2005, vol. 2005, pp. 109–120.

[60] M. Salehie and L. Tahvildari, “Towards a goal-driven approach to action selection in self-adaptive software,” *Softw. Pract. Exp.*, vol. 42, no. 2, pp. 211–233, Feb. 2012.

[61] J. O. Kephart and D. M. Chess, “The vision of autonomic computing,” *Computer (Long. Beach. Calif).*, vol. 36, no. 1, pp. 41–50, Jan. 2003.

[62] C. Krupitzer, F. M. Roth, S. Vansyckel, G. Schiele, and C. Becker, “A survey on engineering approaches for self-adaptive systems,” *Pervasive Mob. Comput.*, vol. 17, no. PB, pp. 184–206, 2015.

[63] H. H. Olsson and J. Bosch, “From opinions to data-driven software R&D: A multi-case study on how to close the ‘open loop’ problem,” *Proc. - 40th Euromicro Conf. Ser. Softw. Eng. Adv. Appl. SEAA 2014*, pp. 9–16, 2014.

[64] P. Dmitriev, B. Frasca, S. Gupta, R. Kohavi, and G. Vaz, “Pitfalls of long-term online controlled experiments,” *Proc. - 2016 IEEE Int. Conf. Big Data, Big Data 2016*, pp. 1367–1376, 2016.

[65] T. Crook, B. Frasca, R. Kohavi, and R. Longbotham, “Seven pitfalls to avoid when running controlled experiments on the web,” *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discov. data Min. - KDD ’09*, p. 1105, 2009.

[66] P. Koomen, L. Pekelis, and D. Walsh, “Peeking at A / B Tests,” pp. 1517–1525, 2017.

[67] A. Deng, J. Lu, and J. Litz, “Trustworthy Analysis of Online A/B Tests,” *Proc. Tenth ACM Int. Conf. Web Search Data Min. - WSDM ’17*, pp. 641–649, 2017.

[68] E. Claeys, P. Gançarski, M. Maumy-Bertrand, and H. Wassner, “Regression Tree for Bandits Models in A/B Testing,” *16th International Symposium on Intelligent Data Analysis, IDA 2017*, vol. 10584 LNCS. Springer Verlag, Strasbourg University, CNRS, ICUBE, 300 Bd Sébastien Brant, Illkirch-Graffenstaden, France, pp. 52–62, 2017.

[69] D. N. Hill, H. Nassif, Y. Liu, A. Iyer, and S. V. N. Vishwanathan, “An Efficient Bandit Algorithm for Realtime Multivariate Optimization,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1813–1821.

[70] B. Kitchenham and S. Charters, “Guidelines for performing Systematic Literature Reviews in Software Engineering,” *Engineering*, vol. 2, p. 1051, 2007.

[71] P. Runeson and M. Höst, “Guidelines for conducting and reporting case study research in software engineering,” *Empir. Softw. Eng.*, vol. 14, no. 2, pp. 131–164, 2009.

[72] R. K. Yin, *Case study research: Design and methods*, vol. 5. London: SAGE, 2009.

[73] C. Robson and K. McCartan, *Real World Research*, 4th ed. John Wiley & Sons Ltd, 2016.

[74] C. R. Kothari, *Research Methodology: Methods and Techniques*, 2nd ed. New Delhi: New Age Internatinal, 2004.

[75] T. H. Davenport, “Smart Business Experiments,” *Harv. Bus. Rev.*, vol. 87, no. 2, pp. 68–76, 2009.

[76] Y. Xu, N. Chen, A. Fernandez, O. Sinno, and A. Bhasin, “From Infrastructure to Culture: A/B Testing Challenges in Large Scale Social Networks,” *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, no. Figure 1, pp. 2227–2236, 2015.

[77] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, and D. Dennison, “Hidden Technical Debt in Machine Learning Systems,” *Nips*, pp. 2494–2502, 2015.

[78] E. Breck, S. Cai, E. Nielsen, M. Salib, and D. Sculley, “What ’ s your ML Test Score ? A rubric for ML production systems,” no. Nips, 2016.

[79] Y. Brun, G. Di Marzo Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Müller, and M. Shaw, “Engineering Self-Adaptive Systems through Feedback Loops.”

[80] M. Salehie and L. Tahvildari, “Self-adaptive software: Landscape and Research Challenges,” *ACM Trans. Auton. Adapt. Syst.*, vol. 4, no. 2, pp. 1–42, 2009.

[81] J. Dowling and V. Cahill, “Self-managed Decentralised Systems Using K-components and Collaborative Reinforcement Learning,” *Proc. 1st ACM SIGSOFT Work. Self-managed Syst. - WOSS ’04*, pp. 39–43, 2004.

[82] IBM, “Autonomic Computing White Paper: An Architectural Blueprint for Autonomic Computing,” *IBM White Pap.*, no. June, p. 34, 2005.

[83] D. Fisch, E. Kalkowski, and B. Sick, “Collaborative Learning by Knowledge Exchange,” in *Organic Computing — A Paradigm Shift for Complex Systems*, no. July 2006, Basel: Springer Basel, 2011, pp. 267–280.

[84] D. Garlan, S.-W. Cheng, A.-C. Huang, B. Schmerl, and P. Steenkiste, “Rainbow: Architecture-Based Self Adaptation with Reusable Infrastructure,” *IEEE Comput.*, vol. 37, no. 10, pp. 46–54, 2004.

[85] J. Kramer and J. Magee, “Self-Managed Systems: an Architectural Challenge,” in *Future of Software Engineering (FOSE ’07)*, 2007, pp. 259–268.

[86] P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimhigner, G. Johnson, N. Medvidovic, A. Quilici, D. S. Rosenblum, and A. L. Wolf, “An architecture-based approach to self-adaptive software,” *IEEE Intell. Syst. their Appl.*, vol. 14, no. 3, pp. 54–62, 1999.

[87] F. Kon, M. Román, P. Liu, J. Mao, T. Yamane, C. Magalhã, and R. H. Campbell, “Monitoring, Security, and Dynamic Configuration with the dynamicTAO Reflective ORB,” *IFIP/ACM Int. Conf. Distrib. Syst. platforms*, pp. 121–143, 2000.

[88] L. Capra, W. Emmerich, and C. Mascolo, “CARISMA: Context-aware reflective mlddleware system for mobile applications,” *Softw. Eng. IEEE Trans.*, vol. 29, no. 10, pp. 929–945, 2003.

[89] T. Patikirikorala, A. Colman, J. Han, and L. Wang, “A Systematic Survey on the Design of Self-Adaptive Software Systems Using Control Engineering Approaches,” *2012 7th Int. Symp. Softw. Eng. Adapt. Self-Managing Syst.*, pp. 33–42, 2012.

[90] D. A. Menascé, H. Gomaa, S. Malek, and J. P. Sousa, “Sassy: A framework for self-architecting service-oriented systems,” *IEEE Softw.*, vol. 28, no. 6, pp. 78–85, 2011.

[91] G. Di Marzo Serugendo, J. Fitzgerald, and A. Romanovsky, “MetaSelf – An Architecture and a Development Method for Dependable Self-* Systems,” *Proc. 2010 ACM Symp. Appl. Comput. - SAC ’10*, pp. 457–461, 2010.

[92] V. Cardellini, E. Casalicchio, V. Grassi, S. Iannucci, F. Lo Presti, and R. Mirandola, “MOSES: A framework for qos driven runtime adaptation of service-oriented systems,” *IEEE Trans. Softw. Eng.*, vol. 38, no. 5, pp. 1138–1159, 2012.

[93] S. Hallsteinsen, K. Geihs, N. Paspallis, F. Eliassen, G. Horn, J. Lorenzo, A. Mamelli, and G. A. Papadopoulos, “A development framework and methodology for self-adapting applications in ubiquitous computing environments,” *J. Syst. Softw.*, vol. 85, no. 12, pp. 2840–2859, 2012.

[94] G. Tesauro, D. M. Chess, W. E. Walsh, R. Das, A. Segal, I. Whalley, J. O. Kephart, and S. R. White, “A multi-agent systems approach to autonomic computing,” *Proc. Third Int. Jt. Conf. Auton. Agents Multiagent Syst. (AAMAS ’04)*, pp. 464–471, 2004.

[95] C. Ballagny, N. Hameurlain, and F. Barbier, “MOCAS: A state-based component model for self-adaptation,” *SASO 2009 - 3rd IEEE Int. Conf. Self-Adaptive Self-Organizing Syst.*, pp. 206–215, 2009.

[96] A. Elkhodary, N. Esfahani, and S. Malek, “FUSION : A Framework for Engineering Self-Tuning Self-Adaptive Software Systems,” *Found. Softw. Eng.*, pp. 7–16, 2010.

[97] M. Reichenbach, R. Seidler, D. Fey, and B. Pfundt, “Organic Computing — A Paradigm Shift for Complex Systems,” *Work*, no. July 2006, pp. 179–192, 2011.

[98] H. J. Goldsby, P. Sawyer, N. Bencomo, B. H. C. Cheng, and D. Hughes, "Goal-based modeling of Dynamically Adaptive System requirements," *Proc. - Fifteenth IEEE Int. Conf. Work. Eng. Comput. Syst. ECBS 2008*, pp. 36–45, 2008.

[99] K. Angelopoulos, V. E. Silva Souza, and J. Pimentel, "Requirements and architectural approaches to adaptive software systems: A comparative study," *8th Int. Symp. Softw. Eng. Adapt. Self-Managing Syst.*, pp. 23–32, 2013.

[100] T. van Oosterhout and A. Visser, "A visual method for robot proxemics measurements," *Proc. Metrics Human-Robot Interact. A Work. Third ACM/IEEE Int. Conf. Human-Robot Interact.*, pp. 61–68, 2008.

[101] E. Lindgren and J. Münch, "Raising the odds of success: the current state of experimentation in product development," *Inf. Softw. Technol.*, vol. 77, pp. 80–91, Sep. 2016.

[102] U. Eliasson, R. Heldal, E. Knauss, and P. Pelliccione, "The need of complementing plan-driven requirements engineering with emerging communication: Experiences from Volvo Car Group," *2015 IEEE 23rd Int. Requir. Eng. Conf. RE 2015 - Proc.*, pp. 372–381, 2015.

[103] J. Bosch and H. H. Olsson, "Climbing the 'Stairway to Heaven'; Evolving From Agile Development to Continuous Deployment of Software," *Contin. Softw. Eng.*, vol. 9783319112, pp. 15–27, 2014.

[104] F. Giaimo and C. Berger, "Design Criteria to Architect Continuous Experimentation for Self-Driving Vehicles," in *2017 IEEE International Conference on Software Architecture (ICSA)*, 2017, no. Icsa, pp. 203–210.

[105] D. I. Mattos, J. Bosch, and H. H. Olsson, "Your System Gets Better Every Day You Use It: Towards Automated Continuous Experimentation," in *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2017, pp. 256–265.

[106] D. I. Mattos, J. Bosch, and H. Holmström Olsson, "More for Less: Automated Experimentation in Software-Intensive Systems," in *The 18th International Conference on Product-Focused Software Process Improvement*, 2017, pp. 146–161.

[107] J. Bosch and U. Eklund, "Eternal embedded software: Towards innovation experiment systems," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7609 LNCS, no. PART 1, pp. 19–31, 2012.

[108] R. Kohavi, A. Deng, R. Longbotham, and Y. Xu, "Seven rules of thumb for web site experimenters," *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. data Min. - KDD '14*, pp. 1857–1866, 2014.

[109] Optimizely, "Optimizely." [Online]. Available: https://www.optimizely.com/. [Accessed: 28-Jun-2017].

[110] H. H. Olsson and J. Bosch, "So Much Data ; So Little Value : A multi-case study on improving the impact of data-driven development practices," in *In Proceedings of the Ibero American Conference on Software Engineering (ClbSE), May 22nd – 23rd, Buenos Aires, Argentina.*, 2017.

[111] B. Zhang, "Privacy Concerns in Online Recommender Systems : Influences of Control and User Data Input," pp. 159–173, 2014.

[112] H. Holmström Olsson and J. Bosch, “From ad hoc to strategic ecosystem management: the Three-Layer Ecosystem Strategy Model? (TeLESM),” *J. Softw. Evol. Process*, no. March, p. e1876, 2017.

[113] A. Fabijan, *Developing the right features: the role and impact of customer and product data in software product development*. 2016.

[114] P. Dmitriev and X. Wu, “Measuring Metrics,” *Proc. 25th ACM Int. Conf. Inf. Knowl. Manag. - CIKM ’16*, pp. 429–437, 2016.

[115] G. Schermann, J. Cito, and P. Leitner, “Continuous Experimentation - Challenges, Implementation Techniques, and Current Research,” *IEEE Softw.*, pp. 1–1, 2018.

[116] L. Li, W. Chu, J. Langford, and R. E. Schapire, “A Contextual-Bandit Approach to Personalized News Article Recommendation,” *Www 2010*, p. 10, 2010.

[117] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley, “Google Vizier,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD ’17*, 2017, pp. 1487–1495.

[118] X. Shang, E. Kaufmann, and M. Valko, “Hierarchical Bandits for "Black Box " Optimization,” Lille, 2015.

[119] Y. Wang, J.-Y. Audibert, and R. Munos, “Algorithms for infinitely many-armed bandits,” *Adv. Neural Inf. Process. Syst.*, pp. 1–8, 2008.

[120] S. Bubeck, R. Munos REMIMUNOS, G. Stoltz, and C. Szepesvári, “X -Armed Bandits,” *J. Mach. Learn. Res.*, vol. 12, pp. 1655–1695, 2011.

[121] G. L. Urban, G. (Gui) Liberali, E. MacDonald, R. Bordley, and J. R. Hauser, “Morphing Banner Advertising,” *Mark. Sci.*, vol. 33, no. 1, pp. 27–46, Jan. 2014.

[122] L. Li, W. Chu, J. Langford, and R. E. Schapire, “A Contextual-bandit Approach to Personalized News Article Recommendation,” in *Proceedings of the 19th International Conference on World Wide Web*, 2010, pp. 661–670.

[123] A. Savitzky and M. J. E. Golay, “Smoothing and Differentiation of Data by Simplified Least Squares Procedures,” *Anal. Chem.*, vol. 36, no. 8, pp. 1627–1639, 1964.

[124] N. Juristo and A. M. Moreno, *Basics of Software Engineering Experimentation*, vol. 5/6. 2001.

[125] X. He, S. Bowers, J. Q. Candela, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, and R. Herbrich, “Practical Lessons from Predicting Clicks on Ads at Facebook,” *Proc. 20th ACM SIGKDD Conf. Knowl. Discov. Data Min. - ADKDD’14*, pp. 1–9, 2014.

[126] R. Kohavi, A. Deng, B. Frasca, T. Walker, Y. Xu, and N. Pohlmann, “Online controlled experiments at large scale,” in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’13*, 2013, p. 1168.

[127] A. Hern, “Why Google has 200m reasons to put engineers over designers,” *The Guardian*, 2014.

[128] J. W. Creswell, *Educational research: Planning, conducting, and evaluating quantitative and qualitative research*, vol. 4. 2012.

[129] S. L. Scott, “Google Analytics Help page,” 2017. [Online]. Available: https://support.google.com/analytics/answer/2844870?hl=en&ref_topic=1745207. [Accessed: 07-Nov-2017].

[130] C. Stucchio, “Saturday is not Tuesday,” 2015. [Online]. Available: https://www.chrisstucchio.com/blog/2015/dont_use_bandits.html.

[131] S. Guha, K. Munagala, and M. Pal, “Multiarmed Bandit Problems with Delayed Feedback,” *arXiv.org*, 2010.

[132] R. Kohavi and R. Longbotham, “Online Controlled Experiments and A/B Tests,” *Encycl. Mach. Learn. Data Min.*, no. Ries 2011, pp. 1–11, 2015.

[133] R. A. Kievit, W. E. Frankenhuis, L. J. Waldorp, and D. Borsboom, “Simpson’s paradox in psychological science: A practical guide,” *Front. Psychol.*, vol. 4, no. AUG, pp. 1–14, 2013.

[134] L. Li, “Offline Evaluation and Optimization for Interactive Systems,” in *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, 2015, pp. 413–414.

[135] M. M. Drugan and A. Nowe, “Designing multi-objective multi-armed bandits algorithms: A study,” in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 2013, pp. 1–8.

[136] J. Brutlag, “Speed Matters,” 2009. [Online]. Available: https://research.googleblog.com/2009/06/speed-matters.html. [Accessed: 09-Nov-2017].

[137] R. Kohavi, R. M. Henne, and D. Sommerfield, “Practical guide to controlled experiments on the web,” *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discov. data Min. - KDD ’07*, vol. 2007, p. 959, 2007.

[138] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, and M. Young, “Machine Learning : The High-Interest Credit Card of Technical Debt,” *NIPS 2014 Work. Softw. Eng. Mach. Learn.*, pp. 1–9, 2014.

[139] X. Wang, Y. Wang, D. Hsu, and Y. Wang, “Exploration in Interactive Personalized Music Recommendation: A Reinforcement Learning Approach,” *ACM Trans. Multimed. Comput. Commun. Appl.*, vol. 11, no. 1, Aug. 2014.

[140] R. Féraud and T. Urvoy, “A stochastic bandit algorithm for scratch games.,” *J. Mach. Learn. Res. Track*, vol. 25, pp. 129–143, 2012.

[141] L. Li, W. Chu, J. Langford, and X. Wang, “Unbiased Offline Evaluation of Contextual-bandit-based News Article Recommendation Algorithms,” *arXiv1003.5956 [cs, stat]*, p. 297, 2011.

[142] K. Jamieson, “FDR Control with Adaptive Sequential Experimental Design,” pp. 1–14, 2017.

[143] T. Kluck and L. Vermeer, “Leaky Abstraction In Online Experimentation Platforms: A Conceptual Framework To Categorize Common Challenges,” 2017.

[144] R. Chen, M. Chen, M. R. Jadav, J. Bae, and D. Matheson, “Faster Online Experimentation by Eliminating Traditional A / A Validation,” pp. 1635–1641, 2017.

[145] R. Kohavi, A. Deng, B. Frasca, T. Walker, Y. Xu, and N. Pohlmann, "Online controlled experiments at large scale," *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discov. data Min. - KDD '13*, p. 1168, 2013.

[146] S. Gupta, S. Bhardwaj, P. Dmitriev, L. Ulanova, P. Raff, and A. Fabijan, "The Anatomy of a Large-Scale Online Experimentation Platform," in *International Conference on Software Architecture ICSA*, 2018, no. May.

[147] I. Gerostathopoulos, C. Prehofer, and T. Bures, "Adapting a System with Noisy Outputs with Statistical Guarantees," *Seams*, 2018.

[148] I. Gerostathopoulos, C. Prehofer, L. Bulej, T. Bures, V. Horky, and P. Tuma, "Cost-Aware Stage-Based Experimentation : Challenges and Emerging Results," in *International Conference on Software Architecture ICSA*, 2018.